64'er

MAI 1984 ÖS 50,–/Sfr 6,– DM 6,–

5|84 DAS MAGAZIN FÜR COMPUTER-FANS

Was bringt Commodores neue Generation?
Test: 264

Lohnt sich ein 7-Farb-Drucker? Test: GP 700 A

Leistungsfähiges Toolkit und Centronics-Interface
Test: KFC Super
Eine nützliche Erweiterung

Der serielle Bus
Alles über die Drucker/Floppy-Schnittstelle

Listing des Monats
Schatzsucher

Großer Vergleichstest
Datenbanken, die es in sich haben

Programmierwettbewerb: 1000 Mark für das beste Kreuzworträtsel

Jede Menge Listings: Zwei Superspiele, Hardcopy mit dem VC 1526, 3D-Grafik und wieder viele Tips und Tricks ★ Kurse: Grafik, Precompiler und Codes.

INHALT

Aktuell

Test

Hardware

Software

Software-Test

Spiele-Test

Programme zum Abtippen

▲ Farbige Hardcopy mit dem GP 700A 24

Die neue Generation: C 264. Was hat er außer einem guten Basic noch zu bieten? 14 ►

◄ Listing des Monats: Schatzsucher. Ein Spi[el] mit Variations- und Ideenreichtum 90

Flipper auf dem Computer. Spielerei oder vollwertiger Ersatz? 60 ►

Datenbanken die es in sich haben. Wir haben uns die besten Datenbanken herausgesucht und getestet, was sich wofür eignet 46

KFC Super — ein Super-Toolkit? Mit diesem Toolkit läßt sich viel Platz sparen 20

Datenbank für jedermann

Heimcomputer lassen sich theoretisch für eine ganze Menge nützlicher Dinge verwenden: Die Briefmarken- oder eine andere Sammlung — sei es nun die Bibliothek oder der Schallplattenbestand — verwalten; Ordnung in die vielen Informationen bringen, die beispielsweise ein Bauherr braucht und so weiter. Wer sich für solche Projekte interessierte, stellte in der Vergangenheit schnell fest, daß er mit einem Heimcomputer nicht weit kam: Die Leistung reichte einfach nicht. Inzwischen ist ein Problem behoben. Floppy-Laufwerke, Vorraussetzung für die effiziente Verwaltung größerer Datenmengen, sind für viele erschwinglich geworden.
Auch für das Softwareproblem zeichnen sich jetzt Lösungen ab: Mit den bislang verfügbaren, relativ simplen Dateiverwaltungsprogrammen. Jetzt kommen für den Commodore 64 eine Reihe von Programmen auf den Markt, für die zwar die Bezeichnung »Datenbank« etwas hochgestochen ist, die aber einer neuen »Generation« zuzuordnen sind. Die besten dieser Programme haben wir getestet — die Berichte finden Sie in dieser Ausgabe. Die Software erlaubt es dem Benutzer eines billigen Computersystems, Arbeiten in Angriff zu nehmen, für die bislang deutlich teurere Anlagen erforderlich waren. Allerdings wird jetzt der Ruf nach einem schnelleren Floppy-Laufwerk noch sehr viel lauter werden.

Michael Pauly, Chefredakteur

Aktuell

Der Marktführer auf dem Mikro-Computer-Sektor, natürlich Commodore, erlebt gegenwärtig einen Auftragsboom, der alle vergleichbaren Vorkommnisse in den Schatten stellt.

Alles im Lot auf dem Commodore-Boot

Der momentane Bestelleingang läßt sogar die zurückliegenden Orders des Weihnachtsgeschäfts nur als Ouvertüre zu einer gewaltigen Oper erscheinen. Das Unternehmen mußte denn auch innerhalb weniger Monate seine Fertigungskapazitäten verdreifachen. Eine weitere Aufstockung erscheint Europachef und Vice President Harald Speyer in absehbarer Zeit fast unausweichlich.
Allein im zweiten Quartal des laufenden Geschäftsjahres (1.10.83 bis 31.12.83) setzte Commodore weltweit insgesamt rund 1,3 Millionen Computer ab; davon wurden 123000 Computer in der Bundesrepublik verkauft. Diese rasante Kapazitätssteigerung erklärt sicherlich die gehäuft auftretenden Qualitätsmängel, kann dies aber nicht rechtfertigen.
Aktuellen Gerüchten, zufolge nach denen sich Commodore in Schwierigkeiten befinde, sind laut Speyer auf »gezielte Neidkampagnen einschlägiger Mitbewerber zurückzuführen« — dem Unternehmen sei es noch nie so gut gegangen wie gegenwärtig. Speyer weiter: »Die kürzlich veröffentlichten Quartals- und Halbjahresergebnisse beweisen eigentlich nur, daß wir erheb

Btx-Anschluß mit Commodore 64

Bildschirmtext wird, entgegen manchen anfänglichen Konzepten und Strategien, für die nächsten Jahre mehr als zusätzliches

Btx ist in diesem Bereich als Alternative zu den konkurrierenden Fernübertragungsmedien (Wähl- oder Standardleitungen, Datex-P) anzusehen, wobei Vorteile wie niedrige Gebühren und vollständige Standardisierung (einziges bis jetzt verwirklichtes sogenanntes »offenes Netz«) gegenüber Anwendungsnachteilen wie begrenztem Komfort und damit begrenzter Produktivität oder langsamen Datenaustausch gegeneinander abzuwägen sind.
Der praktische Nutzen des Bildschirmtexts wächst in starkem Maße, wenn ein Mikrocomputer eingeschaltet wird. Durch die »Intelligenz« des Geräts auf der Benutzerseite wird erst der zweiseitige Austausch von Informationen und Bearbeitungsanweisungen hinsichtlich Geschwindigkeit und Aussagekraft in die Nähe der anderen Datenfernübertragungsmedien gerückt.
Deshalb stellen die Benutzer von Personal- und Homecomputern einen Sonderfall für die Btx-Durchsetzung beziehungsweise Akzeptanz dar: Für das wichtigste Zusatzgerät brauchen sie kein Geld mehr auszugeben. So bezeichnete ein IBM-Sprecher auf dem Online-Kongreß im Februar in Berlin die »Btx-Akzeptanz als Abfallprodukt des wachsenden PC-Einsatzes«.
Der Marktführer für Mikrocomputer sowohl im privaten (Home-) Bereich wie im geschäftlichen (Personal Computer) ist Commodore wie unter anderem das Marktforschungsunternehmen IDC, Wiesbaden, ermittelte. Um dem bisherigen Angebot auf dem deutschen Markt für Btx-fähige Mikro-

Commodore

Die Hamburger Videothek Winterhude bietet einen besonderen Service: Für 60 Mark monatlich kann man dort einen Commodore 64 mit Kassettenrecorder mieten, für 120 Mark gibt es statt dem Recorder ein Diskettenlaufwerk VC 1541 dazu. Die entsprechende Software kann man ebenfalls mieten. Angeboten werden Programme aus den Bereichen Spiele, Lernen und Unterhal

liche Mühe haben, der explodierenden Nachfrage gerecht zu werden«. Im ersten Geschäftshalbjahr stiegen die Umsätze, bezogen auf den entsprechenden Vorjahreszeitraum, international um 129 Prozent auf 640,7 Millionen Dollar. In Deutschland sogar, die Auslandsverkäufe des Werks Braunschweig nicht einbezogen, um 309 Prozent auf 157,3 Millionen Mark. Für das gesamte Geschäftsjahr 1984 erwartet Commodore — vorsichtig geschätzt — einen neuen Umsatzrekord, der international deutlich über einer Milliarde Dollar führen dürfte. Es bleibt nun abzuwarten, ob bei Commodore nach dem Auftragsboom ein Serviceboom folgen wird. (aa)

Die USA-Ecke

Diagnose selbst erstellen

Von Computer Software gibt es jetzt für den Commodore 64 eine Diagnose-Diskette (Kassette). Dieses Programm, 64 Doktor, diagnostiziert Hardwarefehler im Bereich der Tastatur, der Joysticks, des Userports, des Diskettenlaufwerks, des Druckers, des RAM-Speichers, des Kassettenrecorders und von Audio- und Video-Bausteinen. Ein vollständiger Test soll in ungefähr zehn Minuten beendet sein. Um einen defekten Chip zu lokalisieren ist allerdings ein nächtelanger »Dauerlauf« notwendig. Mit dem Videotest kann man unter anderem eine Feineinstellung des Farbspectrums auf dem Monitor vornehmen. 64 Doktor kostet in der Diskettenversion 30 Dollar.

Vom selben Hersteller gibt es auch ein neues Datenbankprogramm, PractiFile, für den Commodore 64. Besondere Kennzeichen: Datenaustausch mit den Tabellenkalkulationsprogrammen PractiCalc und PS (Programmable Spreadsheet) sowie mit mehreren Textverarbeitungsprogrammen soll möglich sein. Der Preis wird sich bei ungefähr 55 Dollar einpendeln. Die Programme sind von Micro Software International, The Skill Mill, 44 Oak Street, Newton, MA 02164 zu beziehen.

Lernen, Sprites generieren, Musizieren

CodePro-64 nennt sich ein Programmpaket von Systems Management mit dem man in die Programmiersprache Basic eingeführt wird (mit Flußdiagrammen auf dem Bildschirm), Sprites generieren kann und Musikstücke komponieren lernt.

Das alles wird grafisch auf dem Bildschirm aufbereitet. So sieht man zum Beispiel seine Kompositionen auf einem schematischen Notenblatt während sie abgespielt werden. Die Auswirkungen bei Befehlsänderungen für die Sprites werden ebenfalls sofort demonstriert. Mitgeliefert wird ein 140 Seiten starkes User Reference Manual. Angeboten wird CodePro-64 von Systems Management Associates, 3700 Computer Drive, P.O. Box 20025, Raleigh, North Carolina 27619.

Medium für Datenfernübertragung im geschäftlichen Verkehr als für private Zwecke oder auch die Kommunikation zwischen Anbietern und Konsumenten Durchsetzung finden — auch mit einem Commodore 64.

computer auch den Niedrigpreisbereich zu erschließen, hat Commodore auf der Hannover Messe '84 einen Btx-Anschluß für Commodore 64 vorgestellt, das einschließlich einem Disketten-Laufwerk für knapp 1400 Mark im Handel angeboten werden soll. Der Btx-Anschluß kostet nur noch rund 250 Mark zusätzlich (genaue Preise sind erst nach Markteinführung gemäß den Kalkulationen des Handels zu erfahren).

Allerdings braucht man für den Anschluß auch noch ein Farbfernsehgerät, das mit einem sogenannten CEPT-Decoder ausgestattet ist; im Handel komplett für knapp 3600 Mark erhältlich.

Der neue Btx-Anschluß zielt auch auf die Anwendung im Unternehmen. Dies ist vor allem unter dem Aspekt zu sehen, daß Btx rechnerisch besonders gut dort abschneidet, wo sehr viele verstreute Stationen angeschlossen werden. Wenn die Kombination von Btx und Computer eine zu hohe Investition erfordert, kann der Gebührenvorteil gegenüber anderen Datenfernverarbeitungsmedien verloren gehen. Die neue Anschlußmöglichkeit kann somit ein adäquates Verhältnis von Geräteinvestition und Gebühren herbeiführen.

Die Kombination von Mikrocomputer und Btx-Fernseher soll in diesem Konzept vor allem die Produktivität des Arbeitsplatzes erhöhen und gleichzeitig die Gebühren senken. Die — verglichen mit anderen Datenübertragungsmethoden — teilweise etwas umständliche Abfrage zum großen Teil vom Commodore 64 automatisiert und damit wesentlich beschleunigt. Außerdem können die Btx-Seiten auf Disk zwischengespeichert und »off-line«, nämlich ohne Verbindung zum Btx-Netz und deshalb auch, ohne daß der Gebührenzähler läuft, bearbeitet werden. Damit geschieht das Ausfüllen der Btx-Antwortseiten ohne Zeitdruck. Die entsprechende Software soll von Commodore geliefert werden. (aa)

64 mieten

tung sowie allgemeine Anwendungen. Alle Programme kann man übrigens — wenn man Gefallen daran gefunden hat — auch einkaufen. Da man hierfür jedoch im Schnitt die sechsfache Leihgebühr hinlegen muß, dürfte der Verkaufserfolg sich wohl in Grenzen halten — es sei denn, in Hamburg hätte man endlich den absoluten Kopierschutz entwickelt. (ev)

Vom Bildschirm auf Kassette?

Ich habe einen Commodore 64 mit Datasette. Ich suche ein Programm, um Daten vom Bildschirm (also Daten, die sich durch irgendeinen Programmablauf ergeben) auf Kassette speichern und dort wieder abrufen zu können. Wer kann mir helfen? Thomas Mandl

Monitor ohne Grünabstufungen?

Kann man an den VC 20 einen S/W-Monitor anschließen, ohne daß bei Mehrplatzbetrieb (verschiedene Vorder- und Hintergrundfarben) unangenehme Grünabstufungen zu sehen sind? (Zum Beispiel durch Verwendung von Pin 1 statt Pin 4 am Monitor-Anschluß). Mirko Wawrowsky

Wie verlängert man Sprites?

Ich besitze einen C 64. Erzeuge ich nun ein Sprite, besteht ja die Möglichkeit, es in X oder Y beziehungsweise in beiden Richtungen zu vergrößern. Das funktioniert bei feststehendem Bewegungsablauf ohne Komplikationen, lege ich jedoch die Koordinaten, an denen das Sprite erscheinen soll, fest, läßt es sich nicht mehr verlängern. Das ist zwar logisch, aber unbefriedigend. Wie muß man vorgehen? Martin Schwarz

Nach einer Stunde keine Farbe mehr?

Bei meinem Commodore 64 passiert es jedesmal, daß er nach zirka einer Stunde keine Farben mehr anzeigt; man sieht nur noch schwarzweiß. Kann man etwas dagegen tun, und wenn ja, was? Klaus Heinz

Können 8032 und 64 zusammenarbeiten?

Ich besitze den CBM 8032 mit Single Floppy CBM 4031 mit Recorder und mit Drucker Epson FX80. Ich möchte gerne einen Grafik-Zusatz kaufen, muß aber feststellen, daß so etwas komplett zirka 1000 Mark kostet (und nur schwarzweiß). Wenn ich in der Auflösung die Ansprüche etwas zurückstelle, so müßte der C 64 doch ein idealer Partner für Farbgrafik sein (wenn man einen geeigneten Farbmonitor kauft oder den heimischen Farbfernseher benutzt). Man hätte den Vorteil, zwei unabhängige Systeme zu besitzen. Wie kann ich diese Commodore-Computer zum Datenaustausch oder Programmaustausch miteinander verbinden? Kann ich von beiden Computern Daten und Programme auf die gemeinsame Floppy übertragen. Alle sagen, daß das im Prinzip möglich ist, aber keiner kann einem genaue Auskunft geben. Vom C 64 auf den Drucker kommt man mit einem V.24-Interface? Kann über dieses Interface auch eine Verbindung zur Single-Floppy CBM-4031 hergestellt werden? Man sagte mir, daß die Daten im C 64 vor dem eigentlichen Ausgang (wenn sie noch parallel sind?) abgegriffen werden müssen. Ich verstehe leider zu wenig davon. Kann man den Commodore 64 als Datenspeicher (eventuell nur für serielle Daten) für den CBM 8032 benutzen? Wie spricht man ihn an? Reine Basic-Programme (ohne PEEKS und POKES) können per Recorder oder per Disk auf beiden Systemen geladen werden und laufen auch. Mit Daten müßte es dann auch möglich sein, und der Zwischenträger — Recorder oder Disk — müßte sich doch auch sparen lassen. Gibt es geeignete Geräte zu kaufen (mit Software?) oder kann man sie selber bauen? Karsten Eckermann

Ist der Drucker eingeschaltet?

Mit welchen Befehlen kann ich den Commodore 64 dazu veranlassen, zu überprüfen, ob der Drucker beziehungsweise die Floppy eingeschaltet ist? Eine Fehlmeldung ließe das Programm abstürzen. Mit vorheriger Überprüfung durch den Computer aber nicht. Ernst Jeschke

VC 20-Programme auf 8032

Wie kann ich Programme vom VC 20 auf dem CBM 8032 zum Laufen bringen? Ich möchte lediglich das Programm über den Drucker auslisten. Gerhard Gruhl

Andere Tastatur für VC 20?

Kann ich an den VC 20 ohne großartige technische Veränderungen eine andere Tastatur, wie zum Beispiel eine CHERRY-Tastatur, anschließen? Wenn ja, können Sie mir auch andere Firmen nennen, die solche Tastaturen vertreiben? Am wichtigsten: Die Tastatur sollte möglichst flach sein, und qualitativ gute Tasten haben. Christoph Dieker

LESER FORUM 64er

MÜNCHEN 8000

Fragen Sie doch!

Selbst bei sorgfältiger Lektüre von Handbüchern und Programmbeschreibungen bleiben beim Anwender immer wieder Fragen offen. Viel mehr Fragen ergeben sich bei Computer-Interessenten, die noch keine festen Kontakte zu Händlern, Herstellern oder Computerclubs haben. Sie können der Redaktion Ihre Fragen schreiben oder Probleme schildern (am einfachsten auf der beigehefteten Karte). Wir veranlassen, daß die Fragen von einem Fachmann beantwortet werden. Allgemein interessierende Fragen und Antworten werden veröffentlicht.

Worin unterscheiden sich Basic-Erweiterungen?

Seit einiger Zeit werden für den VC 20 diverse Spracherweiterungen angeboten. Unter anderem Basic Level 5.5 und Exbasic Level II. Was sind die wichtigsten Unterschiede? Wieviel Speicherplatz belegen diese Erweiterungen in einem VC 20 mit 32 KByte RAM? Rainer Bärwolf

Komma als Satzzeichen?

Ist es möglich, daß man das Komma bei einer Eingabe zu einem INPUT-Befehl beim VC 20 als normales Satzzeichen verwendet, ohne daß der Computer dies, beziehungsweise das nach dem Komma folgende, als zweite Eingabe ansieht und die Fehlermeldung »Extra Ignored« ausgibt? Kann man den Druckerpuffer des SEIKO-GP-100-VC erweitern? Gerhard Giessmann

Beide Fragen lassen sich nur mit »Nein« beantworten.

Was braucht man für Multiplan?

Ich möchte »Multiplan« auf einem Commodore 64 laufen lassen. Welche Commodore-Peripherie benötige ich dazu zusätzlich? Vorhanden: Commodore 64, VC 1541 Floppy Disk, VC 1521 Drucker. Günther Klimek

Multiplan gibt es mittlerweile direkt für das Diskettenformat 1541 (also ohne den Umweg über CP/M). Erhältlich ist Multiplan (in deutsch) bei Happy Software für 336 Mark.

Wie kann man Basic erweitern?

Ich würde gerne den freien RAM-Bereich C000-CFFF beim Commodore 64 dazu nützen, neue Basicbefehle mittels Maschinensprache zu erzeugen. (Zum Beispiel: INSTR, AUTO, RENUMBER, PRINT USING) etc. Könnten Sie mir bitte Quellen angeben, die mir diesbezüglich eine Hilfestellung geben? Ewald Drexler

Bildschirm horizontal scrollen?

Ich besitze einen Commdore 64 mit Floppy-Disk. Um Spiele effektiv zu programmmieren, sind POKEs und PEEKs leider zu langsam. Daher meine Frage: Kann ich mit dem C 64 den Bildschirm horizontal scrollen lassen? Andreas Linz

Welches VC 20-Programm erzeugt Sprache?

Ich bin Besitzer eines VC 20 mit einer 3 KByte-Erweiterung. Wer kennt ein Programm zur Erzeugung von Sprache (Deutsch oder Englisch) auf dem VC 20? Georg Brandt

VC 20: immer nur Basic?

Was gibt es für den VC 20 an anderen Programmiersprachen außer Basic, Forth, Pascal? Wer hat mit diesen Programmiersprachen auf dem VC 20 Erfahrungen gesammelt und kann sie im Leserforum mitteilen? In erster Linie würde natürlich interessieren, welche Versionen von welchen Anbietern sich besonders bewährt oder nicht bewährt haben und für welche Zwecke — außer zur Einarbeitung — sich die eine oder andere Sprache in Verbindung mit dem VC 20 besonders eignet.

Peter Nießen

Wollen Sie antworten?

Wir veröffentlichen auf dieser Seite auch Fragen, die sich nicht ohne weiteres anhand eines guten Archivs oder aufgrund der Sachkunde eines Herstellers beziehungsweise Programmierers beantworten lassen. Das ist vor allem der Fall, wenn es um bestimmte Erfahrungen geht oder um die Suche nach speziellen Programmen beziehungsweise Produkten. Wenn Sie eine Antwort auf eine hier veröffentlichte Frage wissen — oder eine andere bessere Antwort als die hier gelesene — dann schreiben Sie uns doch. Antworten publizieren wir in einer der nächsten Ausgaben. Bei Bedarf stellen wir auch den Kontakt zwischen Lesern her.

Wie wertet man Basic-Programme aus?

Im Data-Becker-Buch »64 intern« ist eine Betriebssystemroutine aufgeführt (Adresse: $AD9F), die beliebige Ausdrücke aus dem Basic-Programm auswertet. Wenn ich diese auf meinem Commdore 64 vom Maschinenprogramm aus aufrufe, macht der Computer alles einwandfrei, bringt aber immer nach dem dritten Aufruf: ?FORMULAR TOO COMPLEX Error. Wie kann man das beheben? *Steffen Roehn*

12-Volt-Betrieb für 64?

Ich suche bisher vergeblich nach einem Anschlußgerät für eine 12-Volt-Batterie zum Commodore 64. Obwohl nach dem vorgeschalteten Transformator zu urteilen der 64 mit 12 Volt betriebsfähig sein müßte.

Walter Wehrenberg

Der C 64 benötigt 5-V-Gleich- und 12-V-Wechselspannung. Diese erhält er über das Netzteil. Er kann daher ohne großen Aufwand auch über eine handelsübliche 12-V-Autobatterie betrieben werden. Schon mit einer 36-Ah-Batterie ist ein einwandfreier Betrieb von mehr als 15 Stunden gewährleistet.

Es sind nur einige einfache Eingriffe auf der Platine nötig, um den C 64 über einen Adapter (der für gleichbleibende Spannungen sorgt), an eine Autobatterie anzuschließen. Als Eingang kann die eingebaute Diodenbuchse benutzt werden.

Die ausführliche Bauanleitung (Beschreibung der Eingriffe auf der Platine und Bauanleitung für den Adpater) ist für 20 Mark bei mir erhältlich, der komplette Adapter (mit Diodenstecker für den C 64 sowie Anschlußklemmen für die Batterie) für 100 Mark. Bitte Verrechnungsscheck beifügen. Meine Anschrift: Dr. Wilfried Herget, Sielkamp 52, 3300 Braunschweig.

Dr. Wilfried Herget

Spielregeln

Wir verschicken keine Prospekte oder ähnliche Produktinformationen — die müssen Sie direkt beim Lieferanten des Produktes anfordern; die Anschrift kann bei uns erfragt werden.

Wir können keine Programme umschreiben oder anpassen. Wenn ein Leser ein von uns veröffentlichtes Programm umgeschrieben hat und bereit ist, das Listing abzugeben, können wir einen entsprechenden Hinweis im Leserforum veröffentlichen.

Ob und wann Antworten auf die veröffentlichten Fragen eingehen, läßt sich nicht voraussagen; wir sind nicht in der Lage, Vormerklisten zu führen und einzelne Leser individuell zu informieren, wenn eine Antwort eingegangen ist. Wir sind aber gern bereit, den Kontakt zwischen verschiedenen Lesern herzustellen, die am gleichen Thema interessiert sind.

Forth-Handbuch auf deutsch?

Ich habe mir zu meinem VC 20 das Modul »Forth« von Audiogenie gekauft. Das Handbuch zu der Programmiersprache ist leider nur in englisch abgefaßt. Kann mir jemand mit einer deutschen Fassung oder mit einigen Tips weiterhelfen?

Hubert Dieterich

Ein gutes Forth-Handbuch in deutscher Sprache gibt es zum Beispiel beim Hofacker, ISBN 3-911682-88-6.

Programm fortsetzen nach Disk-Error

Das Auslesen des Fehlerkanals reicht oft nicht aus, um ein einwandfreies Weiterarbeiten zu ermöglichen. Wie teile ich der Floppy mit, daß der Fehler erkannt wurde und das Programm weiterarbeiten kann?

(Guido Enger)

Immer, wenn man in einem Programm einen Zugriff auf die Diskette macht, muß ein Datenkanal geöffnet werden. Zum Beispiel OPEN 2,8,2, "name,S.W" öffnet eine seqientielle Datei zum Schreiben. Hinter jedem OPEN-Befehl sollte man den Fehlerkanal der Diskette abfragen. Das macht man am besten durch einen Sprung in ein Unterprogramm, das zum Beispiel in Zeile 20000 steht.

```
1190 ...
1200 CLOSE 2: OPEN 2,8,2,"DATEINAME,S,R"
1210 GOSUB 20000 : REM FEHLERKANAL LESEN
1220 IF A1 <> 0 THEN 1200
1230 ...
```

Dieser Programmausschnitt versucht, eine sequentielle Datei mit dem Namen DATEINAME zum Lesen von Daten (Zeile 1200) zu öffnen. Zeile 1210 springt in das Unterprogramm zum Lesen des Fehlerkanals. Nachdem dieser abgearbeitet ist, wird in Zeile 1220 gefragt, ob ein Fehler vorlag (dann war A1 <> 0). In diesem Fall wird ein neuer Versuch gestartet (CLOSE 2 schließt den vorher geöffneten Kanal). Sonst wird das Programm in Zeile 1230 fortgesetzt.

```
20000 REM LESEN DES DISKETTENFEHLERKANALS
20010 :
20020 OPEN 15,8,15 : INPUT#15,A1,A2$,A3,A4
20030 IF A1 = 0 THEN CLOSE 15 : RETURN
20040 PRINT CHR$(147) : REM CLR SCREEN
20050 PRINT A1,A2$,A3,A4 : REM FEHLER WIRD ANGEZEIGT
20060 PRINT »BITTE BESSERN SIE DEN FEHLER AUS«
20070 PRINT " ... UND DRUECKEN SIE >F< ..."
20080 GET R$ : IF R$ <> "F" THEN 20080
20090 RETURN
```

Dieses kleine Programm liest den Fehlerkanal der Diskette. Wenn kein Fehler existiert, ist A1 = 0 und das Programm wird mit RETURN (Zeile 20030) fortgesetzt. Falls ein Fehler existiert, wird er mit Zeile 20050 angezeigt. Jetzt hat man die Möglichkeit, den Fehler auszubessern. Nachden man das getan hat, drückt man F und das Programm wird fortgesetzt. Nicht jeder Diskettenfehler kann behoben werden, ohne Änderungen im Programm vorzunehmen. Aber wenn das Programm O.K. ist, dann handelt es sich in der Regel um, in dieser Form, behebbare Fehler.

Schachprogramme

Ich besitze einen Commodore 64 und entsprechende Software, unter anderem auch die Schachprogramme Sargon II, Petchess und Grandmaster. Da ich ein Vereinsspieler bin (SK Anderssen 2900 Wuppertal), suche ich ein sehr starkes Schachprogramm für den 64er. Die drei oben genannten Schachprogramme sind leider viel zu schwach für mich. Frage: Kennt jemand ein spielstarkes Schachprogramm für den C 64?

(Peter Jugl)

Ersatz für 1541?

Wie kann ich mein Floppy-Disk-Laufwerk von Commodore durch Hard- oder Software-Änderung schneller machen?
(Uwe Busch)

Eine praktikable Methode dafür ist uns nicht bekannt.

Wie kann ich ein BASF-Floppy-Laufwerk 6108 an der 64 anschließen?
(Peter Bosse)

Gibt es eine Möglichkeit, handelsübliche Floppy-Laufwerke beispielsweise von Teac, Sony oder BASF an den 64 anzuschließen? Wo könnte man Baupläne sowie Unterlagen über das Betriebssystem bekommen?
(Heinz Sigrist)

Commodore-Laufwerke verfügen über eigene »Intelligenz«, handelsübliche Laufwerke sind dagegen »dumm«, so daß für ihren Einsatz, abgesehen von einem Betriebssystem, noch ein an den 64 angepaßter Controller nötig wäre. Ein Selbstbau dürfte sich kaum lohnen; unabhängige Anbieter Commodore-kompatibler Laufwerke beziehungsweise Controller sind uns nicht bekannt.

Wordpro 3+ mit 1541?

Ich besitze einen 64 mit Floppy-Laufwerk 1541. Als Textverarbeitungsprogramm wollte ich Wordpro 3+ einsetzen. Dieses Programm scheint aber nur mit Druckern zu arbeiten, die über den IEC-Bus angesteuert werden. Wer weiß wie ein eigener Druckertreiber in Wordpro eingebunden werden kann (Centronics-Schnittstelle über User-Port)? Eine andere Möglichkeit wäre die Umwandlung IEC-seriell/parallel. Wer hat so etwas schon realisiert?
(Heinzjosef Erken)

Es gibt 2 Versionen des Wordpro 3+ für den C 64. Eine für den VC 1526, die andere für andere (Commodore-)Drucker.

Interface für Epson RX-80

Wer bietet ein Interface für den Anschluß eines Epson RX-80 FM an den seriellen Port des VC 20 an, bei dem sowohl die ASCII-Zeichen als auch die Grafiksymbole und Cursorsteuerzeichen des Computers übertragen wurden und vom Drucker richtig verarbeitet werden können?
(Karl Schmaderer)

Ein entsprechendes Interface bieten verschiedene Firmen an, zum Beispiel Microdex in 8036 Herrsching oder Mirwald Electronic in 8025 Unterhaching. Da die Preise zum Teil sehr stark differieren, empfiehlt sich ein aufmerksamer Preisvergleich. Beachten Sie in diesem Zusammenhang auch unsere Rubrik »Expansions«.

Grafik mit VC 1515

Mein Drucker VC 1515 steigt beim Druck von Grafik ständig mit der Fehlermeldung »Gerät nicht vorhanden« aus. Frage: Gibt es eine Software- oder eine Hardwarelösung?
(Joachim Bolle)

Wenn der Drucker im normalen Textmodus zufriedenstellend arbeitet, ist die Wahrscheinlichkeit eines Hardware-Fehlers sehr gering. Überprüfen Sie nochmals Ihr Programm: Wird der Druckerkanal richtig eröffnet (zum Beispiel »OPEN 1,4«) und stimmt die am Selbsttestschalter eingestellte Geräteadresse mit der im Programm verwendeten überein? Ist der Drucker richtig angeschlossen und eingeschaltet? Denken Sie auch daran, daß die Abkürzung »?« für den PRINT-Befehl bei »PRINT#« nicht zulässig ist.

Hardcopy mit VC 1515

Die im Grafikdrucker VC 1515 ausgedruckte Hardcopy läuft nur in der Grundversion. Ich habe eine Datei mit 16 KByte Speichererweiterung und möchte gerne einen Bildschirmausdruck bekommen. Vielleicht können Sie mir die Hardcopy ändern, damit ich meine Datei nicht vollkommen neu aufbauen und eingeben muß.
(Gerd Blinde)

Die Anfangsadresse des Video-RAM ist beim VC 20 leider je nach Ausbaustufe verschieden:
Bis 3 KByte Erweiterung: Video-RAM ab $1E00 (dezimal 7680), bei Erweiterungen > 3 KByte: Video-RAM ab $1000 (dezimal 4096).
Ein Hardcopy-Programm muß natürlich auch auf diese verschiedenen Video-Bereiche zugreifen und im Falle einer Speichererweiterung entsprechend modifiziert werden.

Grafik auf GP 100 VC?

Ich besitze einen VC 20 und den Seikosha GP100VC-Drucker. Wie kann ich die mit dem VC 20 erstellten Grafiken (mit Modul VIC1211A) auf Papier bringen? Woher kann ich eine deutsche Anleitung für den Drucker Seikosha GP100 VC bekommen?
(Andreas Flügge)

Der GP100 VC ist voll grafikfähig, das heißt, die Nadeln können per Software einzeln gesetzt werden. Damit ist natürlich auch die Hardcopy einer Bildschirmgrafik möglich. Man braucht nur ein entsprechendes Programm dafür. Falls einer unserer Leser ein solches bereits geschrieben hat, werden wir es gerne veröffentlichen. Ein deutsches Handbuch zum GP100 VC erhalten Sie bei Ihrem Seikosha-Fachhändler.

Zeilenvorschub steuern?

Der Drucker VC 1515 zu meinem VC 20 läßt im normalen Druckbetrieb zwischen zwei Zeilen zirka 2 mm Platz. Kann ich den Drucker über Software so steuern, daß zwischen den Zeilen kein Zwischenraum bleibt? Anders wäre eine Grafik auf dem Drucker unmöglich, da zum Beispiel eine Kurve in Zeilen aufgeteilt würde.
(Henner Köhne)

Der Zeilenvorschub wird beim VC 1515 automatisch durch den gewählten Zeichenmodus gesteuert. Wird mit BS-Code in den Grafik-Modus geschaltet, dann wird die Ausgabe eines Zeilenzwischenraumes unterdrückt, um ordentliche Grafiken darstellen zu können.

Kleinschrift auf VC 1515

Ich besitze einen VC 20 mit Drucker VC 1515. Wie kann ich ein Programm-Listing im Kleinschriftmodus abdrucken?
(Martin Quinke)

Der VC 1515-Drucker wird mit folgender Kommandozeile zur Ausgabe eines Listings im Kleinschriftmodus veranlaßt: »OPEN 4,4,7:PRINT#4,CHR$(17):CMD 4:LIST«. Vorsicht, die Angabe der Sekundäradresse 7 nicht vergessen! Mit »OPEN 4,4:PRINT#4,CHR$(145)« kann man wieder in den Normalmodus zurückschalten (nachzulesen im VC 1515-Handbuch Seite 30/31).

Schreibmaschine als Drucker?

Welche Schreibmaschine verfügt über eine Schnittstelle für den VC 20? *(Harald Küst)*

Prinzipiell lassen sich alle elektrischen Schreibmaschinen, die überhaupt für den Anschluß an einen Computer vorgesehen sind, auch am VC 20 betreiben. In den meisten Fällen ist dazu ein V.24-Interface notwendig, einige Schreibmaschinen lassen sich jedoch auch direkt über den seriellen Bus des VC 20 anschließen, zum Beispiel Brother EP-20C oder Olympia Compact 2. Sie müssen sich allerdings darüber im klaren sein, daß eine Schreibmaschine weder Grafik- noch Steuerzeichen des VC 20 drucken kann.

Entfernungen mit VC 20 berechnen?

Ich bin Besitzer eines VC 20 (16 KByte). Wie kann ich mit Hilfe des im VC 20 definierten Bogenmaßes die Entfernung zwischen zwei Koordinaten auf der Oberfläche der Erde und den Kurs zwischen den beiden Punkten berechnen (zum Beispiel zwischen 30°34'N 10°49 E und 50°45N 13°49'E)?
(Harald Lang)

Hier gibt es Clubs

In Hamburg-Harburg wurde der C 64 User-Club-Harburg gegründet. Nach den Vorstellungen der Gründer sollte er in den Anfängen folgende Ideen verwirklichen:
* Hilfestellung für Anfänger mit dem Computer und Programmen.
* Wissenserweiterung
* gemeinsamer Einkauf
* Übersetzungen von Arbeitsanleitungen und Spielanweisungen
* Tausch von Programmen im Rahmen der gesetzlichen Möglichkeiten.

Ziele, auf die in Zukunft hingearbeitet werden soll, sind:
* Erlernen und Vermitteln von Programmiersprachen
* Einstieg und Anwendung von C/PM-Programmen
* Umgang mit kommerziellen Programmen
* Anpassen von Programmen an die persönlichen Bedürfnisse des Anwenders.

Mitglied kann jeder C 64-Besitzer werden, und zwar auch dann, wenn er zur Zeit noch nicht den Ehrgeiz hat, »Computer-Fachfrau/mann« zu werden.

Die Mitgliedschaft ist kostenlos, bis auf tatsächlich anfallende Auslagen wie Porto, Anzeigen etc.

Der Club trifft sich zirka alle 4 bis 6 Wochen bei einem der Mitglieder.

Ansprechpartner sind: Andreas Gröschl, Lasallestr. 54, 2100 Hamburg 90, Tel. 7651923 und Werner Thöle, Tel. 7540598. Telefonische Anfragen bitte nur in der Zeit von 19.00 bis 20.00 Uhr.

Ein Wolf im Schafspelz – der 264

Bild 3. Der Maschinensprache-Monitor ist schon eingebaut

Bild 1. Commodore 264

Die 64'er Redaktion hatte als erste die Möglichkeit, den Commodore 264 zu testen. Was uns überzeugt, ist das eingebaute Basic. Der zusätzliche Maschinensprache-Monitor rundet das positive Bild von diesem neuen Computer ab.

Bild 2. Die Built-In-Software des C 264

So beliebt und verbreitet der Commodore VC 20 und C 64 auch sind, einige Dinge hat man doch bemängelt, und das zu Recht. Vor allem ist es ihr spartanisches Basic. Befehle, die andere Computer bereits »in sich« haben, mußten beim VC 20 und C 64 »soft« umgangen werden. Und das war natürlich auch die Geburtsstunde für Basicerweiterungen, Toolkits, Assemblern, Grafikmodule und was da sonst noch auf den Markt gekommen ist.

Das alles weiß Commodore natürlich auch. Viele andere Computerhersteller geben Ihren Computern ein wesentlich komfortableres Basic mit. Mit dem neuen C264 (Bild 1) setzt Commodore zwar keinen neuen Standard, aber es wird gleichgezogen. Und berücksichtigt man zum Beispiel den komfortablen Editor, meiner Meinung nach der Beste in dieser Leistungsklasse, übrigens der gleiche wie der des C64, und den eingebauten Maschinensprache-Monitor, Assembler und Disassembler, so findet der 264 seinen Platz in Puncto Bedienungsfreundlichkeit ganz vorne, an der Spitze. Dabei ist die schon in der Ausgabe 4, Seite 9 ff erwähnte Built-In-Software noch gar nicht berücksichtigt.

Built-In-Software

Laut Commodore wird es folgende »eingebaute« Programme geben (Bild 2):

* Magic Desk, eine Simulation eines kompletten Büroschreibtisches, mit den Funktionen Briefe schreiben, Rechnen, Schriftverkehrablage, Zeitanzeige und so weiter, ein Programm mehr für den Home und privaten Einsatz (Wir werden noch einen ausführlichen Test dazu bringen).
* Superscript, eine semiprofessionelle Textverarbeitung
* Commodore 3-Plus-1
* Logo, eine Programmiersprache vor allem für den Anfänger
* Pilot, eine weitere Programmiersprache
* Easycalc 264, ein Tabellenkalkulationsprogramm
* COM 264
* Financial Advisor

Der Commodore 264 wird mit einem dieser Programme, das fest installiert wird, ausgeliefert. Der Käufer gibt bei seiner Bestellung an, welche Software er haben will. Diese Programme lassen sich dann kinderleicht über die Funktionstasten ein- und auch wieder ausschalten.

Alle restlichen oben angegebenen Programme können dann über den Expansionport als Modul eingesteckt und benutzt werden.

Da ein Wechseln der Built-In-Software nicht so einfach möglich sein dürfte (der Computer muß geöffnet, das ROM ersetzt werden, Garantiebedingungen müssen beachtet werden), ist schon beim Kauf des C264 eine wichtige Entscheidung zu treffen. Hoffentlich werden diese Überlegungen durch eine umfassende Beratung durch den Verkäufer und Händler unterstützt. Sobald der C264 mit dieser Software auf dem deutschen Markt erhältlich sein wird, werden wir Ihnen wichtige Informationen und Hilfestellungen geben. Diese Programme werden oder sind nämlich zum Teil auch schon für den C 64 erhältlich.

Doch kommen wir zum Basic des C264: Wie schon kurz angedeutet, besitzt der Commodore 264 ein recht komfortables Basic. Es unterstützt jetzt nicht nur die Tonausgabe, auch umfangreiche und mächtige Grafik-Befehle sowie Hilfsfunktionen, Sound- und Diskettenbefehle wurden zusätzlich zum bisherigen Commodore-Basic implementiert.

Viele Befehle, nämlich alle, die sich im Direkt-Modus ausführen lassen, können über die Funktionstasten programmiert werden. Sie sind frei programmierbar. Das funktioniert genauso wie beim Simons-Basic mit dem Key-Befehl. Die jeweilige Belegung dieser Tasten kann mit der Anweisung DISPLAY abgerufen werden. Eine Besonderheit ist die Help-Funktion. Wurde ein Programm mit einer Fehlermeldung abgebrochen, ist es möglich, sich die fehlerhafte Programmzeile durch Drücken der Help-Taste (zum Beispiel Funktionstaste f8) revers anzeigen zu lassen. Leider wird nicht der Fehler in der Zeile selbst angezeigt, etwa wie bei ExBasic-Level II, wo der Cursor direkt auf den Fehler zeigt. Der Wert dieser Help-Funktion ist eigentlich nicht ganz einsehbar, da auch die normale Basic-Fehlermeldung auf die fehlerhafte Zeile hinweist.

Grafik-Befehle

Es ist möglich, drei Bildschirmmodi darzustellen: Den hochauflösenden Grafik-Modus mit einer Auflösung von 200 x 320 Punkten, den Multicolor-Modus mit einer Auflösung von 160 x 200 Punkten und den normalen Textmodus mit 40 x 25 Zeichen. Diese verschiedenen Modi können, und das ist ein echtes Plus, gleichzeitig dargestellt werden. Sie werden durch den GRAPHIC-Befehl definiert. Und das wird folgendermaßen gelöst:

Modus 0 = Textdarstellung, das ist der Standard-Modus, in dem sich der C264 nach dem Einschalten befindet.
Modus 1 = Hochauflösende Grafik
Modus 2 = Hochauflösende Grafik + Textdarstellung in den unteren fünf Zeilen
Modus 3 = Multicolor Grafik
Modus 4 = Multicolor Grafik + Text in den unteren fünf Zeilen

Es ist auch möglich, im hochauflösenden Grafikbereich Text hineinzusetzen (siehe CHAR-Befehl). Im Modus 3 und 4 kann vollkommen unabhängig vom Hires-Bildschirm, also ohne ihn zu beeinflussen, irgendein Text in den letzten fünf Zeilen des Bildschirms dargestellt werden. Es ist somit möglich, im oberen Teil eine Grafik zu setzen und gleichzeitig unten durch den LIST-Befehl ein Listing ablaufen, oder zum Beispiel eine Tabelle sich anzeigen zu lassen. Durch Wahl der Grafik-Modi 1 bis 4 wird ein Speicherplatz von 10 KByte reserviert. Und mit rund 60 KByte frei programmierbarem RAM-Speicher ist das keine nennenswerte Einschränkung.

Der C264 ist in der Lage, 128 Farben darzustellen. Genauer gesagt, 16 Farben in je acht verschiedenen Abstufungen (Bild 8). Diese Farben können direkt angesprochen werden, entweder mit der Control-Taste im Direkt-Modus oder durch einfachen Basic-Befehl im Programm. Die Grafik-Befehle sehen Sie in Bild 5, 6 und 9.

Bild 4. Ein Beispiel für geschachtelte Schleifen

Bild 5. Die wichtigsten Grafik-Befehle

Bild 7. Windowing beim 264. In jedem Fenster laufen Aktionen unabhängig voneinander ab. ▶

COLOR,	Region, Farbe, Farbintensität
	Region (0 bis 4)
	0 = Hintergrundfarbe
	1 = Zeichenfarbe
	2 = Multicolor 1
	3 = Multicolor 2
	4 = Rahmenfarbe
	Farbe (1 bis 16)
	Farbintensität (0 = dunkel, bis 7 = hell)
GRAPHIC	Modus, löschen
	Modus (0 bis 4)
	0 = Text
	1 = Hires
	2 = Hires + Text (unterteilter Bildschirm)
	3 = Multicolor 1
	4 = Multicolor 2 (unterteilter Bildschirm)
	löschen (0 oder 1)
	0 = nicht löschen
	1 = löschen

◀ Bild 6. Farb- und Grafikmodi

Der C264 hat zum Teil sehr mächtige Grafik-Befehle. Sie können sicher schon abschätzen, wie einfach jetzt Grafik erstellt werden kann. Sicher haben Sie bemerkt, daß keine Rede mehr von Sprites ist. Und in der Tat, die beim C 64 so beliebten Sprites gibt es beim C264 nicht mehr. Man kann sich zwar bestimmte Bereiche des Bildschirms reservieren und auch verschieben, dabei ist man auch nicht beschränkt auf die bei den Sprites begrenzten Größe von 24 x 21 Punkten. Ob jedoch mit Merkmalen wie Kollisionsabfrage, etwas Entsprechendes wie ein Hintergrund- beziehungsweise Vordergrund-Sprite definiert werden kann, mit allen sich daraus ergebenden Möglichkeiten für Spiele, ist zumindest zweifelhaft. Aus dem Basic-Wortschatz und dem Handbuch war das nicht ersichtlich.

Apropos Handbuch: Auch dieses bisher stiefmütterlich behandelte Thema reiht sich in der Reihe der positiven Eindrücke nahtlos ein. Es enthält eine umfassende Erklärung aller Basic-Befehle, mit zahlreichen Beispielen, einer kompletten Liste und Beschreibung der Monitor-Befehle und mehreren Anhängen. Dieses Handbuch läßt sicherlich nicht so viele Fragen offen wie man es bisher von Commodore gewohnt war.

Kommen wir zum nächsten Thema.

Sound-Befehle

Was man beim Basic Positives gemacht hat, ist genau entgegengesetzt den Sound-Möglichkeiten des C264. Es gibt nämlich nur zwei Befehle, die die »musikalischen« Eigenschaften des C264 unterstützen (Bild 10). Mehr sind auch gar nicht notwendig, da die Sound-Fähigkei-

▲ Bild 8. Die 16 Farben des C 264 mit je 8 Abstufungen

▼ Bild 9. Die Grafik-Befehle des 264

CHAR	= fügt Texte direkt in Hires-Grafik ein
BOX	= zeichnet Rechtecke
CIRCLE	= zeichnet Kreise, Ellipsen
PAINT	= füllt einen bestimmten Bereich mit einer bestimmten Farbe (wie in Simons-Basic)
SCALE	= Die Skalierung der Bit-Muster der Hires- und Multicolor-Grafik kann geändert werden.
DRAW	= Zeichnet Punkte und Linien
LOCATE	= setzt den Anfangspunkt eines Zeichenbefehls fest
SSHAPE	= speichert einen zu bestimmenden Teil der Hires Grafik auf Band oder Diskette
GSHAPE	= Diesen Teil kann man mit GSHAPE wieder an irgendeiner Stelle des Grafik-Bildschirms auszeichnen lassen. GSHAPE erlaubt mehrere Modi: Modus (0 bis 4)
	0 = wie gespeichert
	1 = Reverse Darstellung
	2 = OR — verknüpft mit der Umgebung
	3 = AND — verknüpft mit der Umgebung
	4 = XOR — verknüpft mit der Umgebung
RCLR(N)	= Ordnet einer bestimmten Bildschirmzone N (0 bis 4) einer bestimmten Farbe zu (Zonen siehe COLOR)
RDOT(N)	= gibt die momentane Position des Grafik-Cursors an
RGR(N)	= man erhält den Grafik-Modus in dem man sich befindet
RLUM(N)	= gibt die zugewiesene Farbintensität der Farbzone N an

VOL (0 bis 7)	= reguliert die Lautstärke der erzeugten Töne
SOUND x,y,z	= erzeugt einen Ton beziehungsweise ein Geräusch
x (1 bis 3)	
1	= hohe Stimme von sehr hoch bis mitteltief
2	= tiefe Stimme von mittelhoch bis sehr tief
3	= Geräusche (zum Beispiel Donnern)
y (0 bis 1023)	= Notenwert, (0=tiefer, 1023=hoher Ton)
z (0 bis 65535)	= Zeitdauer des erzeugten Tones

Bild 10. Die Sound-Befehle des C 264

ten des C264 gegenüber denen des C 64 stark eingeschränkt wurden.

Ich selbst habe zwischen den beiden Stimmen (x=1 und x=2) allerdings keinen Unterschied feststellen können. Die fantastischen Synthesizer-Eigenschaften des C 64 wurden fast völlig unter den Tisch gekehrt. Ob diese Einschränkung negativ zu bewerten ist, bleibt dahingestellt. Denn Commodore hatte mit diesem Modell nicht vor, seinen Renner C 64 abzulösen.

Dafür können sich die Hilfsfunktionen des C264 wieder sehen lassen. Es gibt vier Hilfs-Funktionen, die ein effektives Editieren von Basic-Programm stark unterstützen (Bild 12)

Eine andere Gruppe sind die Disketten-Befehle. Sie entsprechen denen der CBM 8000er Reihe (Bild 11.)

Ein umständliches Öffnen der Datenkanäle entfällt somit. Allerdings sind die vom C 64 bekannten Disketten-Befehle auch anwendbar. Der Fehlerstatus eines Laufwerkes kann einfach über die Variablen DS und DS$ angezeigt werden. Sie enthalten die Fehlerart und die entsprechende Fehlermeldung.

Aber auch das Basic unterstützt die Fehlererkennung im Programm. Es gibt die Möglichkeit, dem Programm »Fallen« zu stellen, Commodore nennt es »trapping«.

Mit diesen beiden zusammengehörigen Befehlen (TRAP und RESUME, siehe Bild 13) kann das Pro-

DLOAD/DSAVE	= laden/speichern von Programmen
DIRECTORY	= Der Befehl dürfte klar sein. Das Anzeigen des Directorys kann abgebrochen oder verlangsamt werden. Auch eine Auswahl der anzuzeigenden Programme ist möglich (zum Beispiel alle Programme, die mit A beginnen)
COLLECT	= entspricht dem VALIDATE-Befehl beim VC 20/C 64.
COPY	= kopiert ein Programm von einer Diskette auf eine andere bei einem Doppellaufwerk oder kopiert ein Programm innerhalb einer Diskette unter einem anderen Namen beim Einzellaufwerk.
RENAME	= Ändert den Namen eines Files
SCRATCH	= löscht ein File auf einer Diskette

Bild 11. Diskettenbefehle wie bei der 4/8000-Serie

AUTO Abstand	= AUTO 20 erzeugt nach jedem RETURN eine neue Zeilennummer, die um 20 höher ist als die Zeile davor
RENUMBER x,y,z	= der komfortabelste RENUMBER-Befehl, der mir bekannt ist.
x	= neue Anfangszeile
y	= Abstand
z	= ab welcher bisherigen Zeile Der Befehl RENUMBER 1000, 10, 100 nummeriert ein Basic Programm ab der bisherigen Zeile 100 im Abstand von 10 neu durch, wobei die erste neue Zeile mit 1000 beginnt. Alle Sprungadressen (GOTO, GOSUB, ON x GOTO/GOSUB etc.) werden berücksichtigt. Selbst wenn diese Sprungbefehle vor z liegen.
DELETE Bereich	= löscht Programmzeilen im angegebenen Bereich
TRON, TROFF	= schaltet den Trace Modus ein oder aus. Die jeweils bearbeitete Zeile wird angezeigt.

Bild 12. Diese Befehle helfen bei der Programmerstellung

TRAP	= fängt einen Fehler ab. Der Fehler kann mit den Fehlervariablen ER, ERR$ und EL ausgegeben werden. ER = gibt den zuletzt aufgetauchten Fehler aus ERR$ = enthält die entsprechende Basic-Fehlermeldung EL = gibt die Zeile an, in der sich der Fehler befindet
RESUME	= Mit diesem Befehl wird das Programm fortgesetzt, ohne daß der Fehler zum Absturz des Porgramms führt. Hier besteht die Möglichkeit, das Programm an irgendeiner Stelle weiterlaufen zu lassen indem die entsprechende Zeilennummer angegeben wird. Durch RESUME NEXT wird das Programm direkt hinter der fehlerhaften Zeile fortgeführt.

Bild 13. Fehlerbehandlung beim neuen Commodore 264

gramm absturzsicher gemacht werden. Auch wenn man zum Beispiel die Stop-Taste betätigt, gibt es keinen BREAK, sondern das Programm wird nach dem Loslassen der STOP-Taste ohne eine Fehlermeldung fortgesetzt. Nur der Fehler FILE NOT FOUND wird nicht berücksichtigt.

Aber es kommt noch besser. Es wurden Befehle implementiert, die das Strukturierte Programmieren unterstützen (siehe Bild 4 und 15).

Im Bild 16 sind die Befehle zusammengefaßt, die sich in keine bestimmte Kategorie eingliedern lassen

Aus den Bildern erkennen Sie alle Befehle, die das neue Commodore Basic 3.5 bietet. Selbstverständlich kommen noch die vom VC 20/C 64 hinzu. Sie sehen selbst, wie komfortabel es geworden ist. Es läßt kaum noch Wünsche offen. Aber das ist immer noch nicht alles. Wie schon kurz erwähnt, besitzt der Commodore C264 auch noch einen eingebauten Maschinensprache-Monitor. Und das ist eigentlich eine Untertreibung. Denn es gibt nicht nur die sonst üblichen Monitorfunktionen. Auch ein Assemblieren und Disassemblieren ist damit möglich. TEDMON, so wird er genannt, und Basic können nebeneinander laufen und schließen sich nicht gegenseitig aus. Es ist auf einfache Weise möglich, entweder Assemblerprogramme separat oder als Unterprogramm in Basic ablaufen zu lassen. Der TEDMON wird von Basic mit dem Befehl MONITOR aufgerufen. Danach stehen eine ganze Reihe von Funktionen zur Verfügung (siehe Bild 3 und 14).

Betrachtet man das gesamte Gerät mit allen seinen Möglichkeiten wie das vorzügliche Basic, den eingebauten Maschinensprache-Monitor sowie der Built-In-Software, so kann man Commodore zu diesem Computer beglückwünschen. Im Vergleich zum C 64 kann man sagen, daß der C264 zwar in seinen Grafikmöglichkeiten etwas eingeschränkt (keine Sprites) und der Sound-Teil erheblich reduziert wurde, jedoch die Bedienerfreundlichkeit um ein gewaltiges Maß gestiegen ist. Auch das Gehäuse macht einen stabilen Eindruck und ist sehr kompakt. Die Tastatur ist leichtgängig und die Idee mit den Cursortasten (siehe Bild 1) ist auch nicht übel. Diese Merkmale und der Preis von voraussichtlich um 1200 Mark grenzen dann auch den C264 vom C 64 ab. Der C264 geht eindeutig in Richtung Anwendung. Seine Software (Basic Windowingfähigkeit (siehe Bild 7) und Built-In-Software) unterstützt stark die Programmierung und Nutzung profesioneller Software.

Leider setzt auch Commodore, wie so viele europäische und amerikanische Computerhersteller, die unheilvolle Strategie der Inkompatibilitäten fort. Es hätte doch möglich sein müssen, zumindest die Anschlußmöglichkeiten der Peripherie voll kompatibel mit dem C 64 zu machen. (gk)

Befehl	Funktion
A = Assemble	— wandelt ein Assemblerprogramm in Maschinencode um
C = Compare	— vergleicht zwei Sektionen des Speichers und teilt die Untertschiede mit
D = Disassemble	— wandelt Maschinencode um in Assemblerbefehle
F = Fill	— belegt den Speicher mit einem spezifizierten Byte
G = Go	— startet ein Assemblerprogramm ab einer bestimmten Adresse
H = Hunt	— suche den Speicher nach allem Vorkommen eines bestimmten Bytes ab.
L = Load	— lädt ein Programm von Kassette oder Diskette
M = Memory	— gibt die Inhalte von Speicherstellen an
R = Register	— gibt die 6502 Register-Werte an
S = Save	— speichert auf Band oder Diskette
T = Transfer	— transferiert einen Teil des Speichers in einen anderen
X = Exit	— verläßt TEDMON

Bild 14. Alle Funktionen des Maschinensprache-Monitors TEDMON

Befehle	Beschreibung
DO UNTIL DO WHILE EXIT LOOP UNTIL LOOP WHILE IF THEN ELSE	Mit diesen Befehlen ist ein voll strukturierter Programmablauf möglich. Die Endebedingung kann am Anfang der Schleife (DO UNTIL/WHILE) oder an das Ende (LOOP UNTIL/WHILE) gesetzt werden. Durch EXIT kann die Schleife während eines Schleifendurchlaufs verlassen werden.

Bild 15. Auch strukturierte Programmierung ist möglich

Befehl	Funktion
PRINT USING	= Es definiert das Format eines Textstrings oder einer Zahlenreihe bei der Ausgabe. Mit ihm wird das Erstellen von Tabellen jeglicher Art zum reinen Vergnügen.
PUDEF	= ändert Parameter des PRINT USING-Befehls
INSTR	= erlaubt es, zwei Strings ineinander zu verflechten.
DEC	= Umwandlung einer Hexadezimalzahl (00 bis FF) in eine Dezimalzahl.
HEX$	= Umwandlung einer Dezimalzahl (0 bis 65536) in eine Hexadezimalzahl
JOY	= Position der Joysticks 1 oder 2
GET KEY	= ähnelt dem Get-Befehl, wartet jedoch bis eine Taste gedrückt wird.

Bild 16. Noch einige Befehle

KFC-Super – ein Super-Toolkit?

Das KFC-Super ist eine Erweiterung für den Commodore 64, die außer einer großen Anzahl von Toolkit-Befehlen einen zusätzlichen Maschinensprache-Monitor und auch noch eine Centronics-Schnittstelle besitzt. Ist es empfehlenswert?

Jeder Commodore 64-Besitzer weiß spätestens, nachdem er das (magere) Handbuch gelesen hat, daß das Basic des C 64 noch viele Wünsche offen läßt. Auch werden viele Benutzer, die sich nicht nur mit Basic beschäftigen, einen eingebauten Monitor vermissen.

Dieses Problem wurde in der Vergangenheit bereits mehrmals durch nachladbare beziehungsweise einsteckbare Basicerweiterungen zu lösen versucht (Simons Basic, Extended Basic, usw.). Ein Produkt aus diesem Bereich ist das Steckmodul »KFC Super« (Bild 1).

Das Modul wird als offene Platine mit vier EPROM-Steckplätzen, von denen zwei bestückt sind, geliefert.

Bild 1 Die Platine des KFC Super GK mit zwei freien Steckplätze für weitere EPROMs

Auf der Platine befindet sich ferner ein Resetknopf, ein Dip-Schalter und ein Ein/Aus-Schalter. Das Modul ist für eine Aufnahme im Expansion-Port vorgesehen und paßt einwandfrei, auch wenn der Anblick einer offenen Platine nicht jedermanns Geschmack ist.

Die beiliegende Bedienungsanweisung, die leider mit Beispielprogrammen recht spärlich umgeht, verspricht viel: KFC Super sei nicht nur ein Toolkit, sondern auch ein Monitor mit außergewöhnlichen Fähigkeiten.

Nach dem Einstecken des Moduls meldet sich KFC Super mit eigenem Titelbild und Angabe des freien Basicspeichers (30719 Byte). Der Verlust an Speicherplatz für Basic-Programme erklärt sich leicht, wenn man den Speicherbereich betrachtet, in den sich das KFC Super lädt. Es steht im sogenannten Autostartbereich ab $8000 (32768), der Monitor ab $9000 (36864), also im oberen Bereich des Basicspeichers. Glücklicherweise muß man beim C 64 nicht mit jedem Byte geizen, so daß dieser Verlust wahrscheinlich erst beim Einsatz fertiger Programme zum Tragen kommt.

Der Ein/Aus-Schalter sorgt dafür, daß man das Modul nicht ständig heraus- und hereinstecken muß, wenn man zum Beispiel Spiele von Diskette laden will. Auch kann man mittels des Befehls KILL in das normale Basic springen. Durch Drücken der Resettaste kann man das Toolkit wieder reaktivieren. Allerdings wird ein im Speicher befindliches Basic-Programm dann gelöscht.

Die Befehlsliste des KFC Super ist so ausgelegt, daß sie die tägliche Arbeit und das Programmieren erleichtert. Es lassen sich drei Gruppen von Befehlen unterscheiden: Programmierhilfen, Floppy-Befehle und Kassetten-Befehle (Bild 2).

Zusätzlich stehen noch Befehle zur Vereinfachung des Programmablaufes wie PAUSE, ESC, REPEAT und OFF zur Verfügung, die es möglich machen, das Programm anzuhalten und wieder zu starten.

Bis hierhin unterscheidet sich das KFC Super nur wenig von bereits auf dem Markt befindlichen Toolkits. Die eigentliche Besonderheit ist in zwei fest eingebauten Sonderfunktionen zu sehen:

Zum Einen ist eine Centronics Schnittstelle eingebaut. Durch ein einfaches Kabel verbunden (User-Port-Drucker) kann man über Vorwahl der Geräteadresse verschiedene Druckmodi erreichen: Die Geräteadresse 16 bewirkt, daß alle Daten ohne Wandlung in 8-Bit Wortbreite übertragen werden; dieser Modus eignet sich für Grafik. Die Geräteadresse 18 wird als Textmodus bezeichnet, mit ihm kann die Groß/Kleinschreibung des Commodore realisiert werden. Da bei den beiden ersten Modi keine Steuerzeichen beim Auflisten erzeugt werden, kann über die Geräteadresse 20 in den sogenannten Listmodus geschaltet werden. Die Steuerzeichen des Commodore werden dann als Kleinbuchstaben ausgedruckt.

Betrachtet man die Tatsache, daß es sich bei dieser Schnittstelle eigentlich nur um eine Zugabe handelt, ist es doch erstaunlich, wie leistungsfähig sie ist. Leider wurden die Geräteadressen so unglücklich gewählt, daß es bei den meisten fertigen Programmen, die eine Vorwahl der Druckeradresse nicht zulassen, zu keiner Reaktion des Druckers kommen wird. Zweitens belegt das KFC-Super einen sehr wichtigen Speicherbereich, der von vielen Textprogrammen ebenfalls

Bild 2. Der Befehlsvorrat des KFC Super

1. Die Programmierhilfen

Befehl	Funktion
AUTO	gibt automatisch die Zeilennummern vor, die Parameter sind wählbar
RENUM	numeriert alle Zeilen neu, Parameter sind wählbar, GOTO, GOSUB ... wird ebenfalls umnummeriert
DEL	löscht Basic-Zeilen oder Gruppen
TRACE	zeigt die Zeilennummern beim Programmablauf
FIND	sucht nach Zeichen und Begriffen
HELP	zeigt die Stelle, an der ein Programmierfehler auftrat
OLD	rettet nach einem NEW oder RESET das Basic-Programm
DUMP	Zeigt die Variablen und ihren Wert

2. Die Floppy-Befehle

Befehl	Funktion
CATALOG	zeigt das Directory ohne Programmzerstörung
STATUS	Anzeige einer Floppy-Fehlermeldung
DLOAD	Programm von Diskette laden
DVERIFY	Programm mit der Diskette vergleichen
DMERGE	Programm an bestehendes anhängen
DSAVE	Programm auf Diskette speichern
DISK	Vereinfachte Befehlsübergabe an die Floppy (N,V,R,I,C,S)

3. Die Kassetten-Befehle

Befehl	Funktion
PUT	Schnelles Speichern auf Kassette
GET	Schnelles Laden von Kassette
COMP	Schneller Verify von Kassette
MERGE	Programm von Kassette anhängen

benutzt wird, so daß die Zusammenarbeit mit einigen Textverarbeitungsprogrammen sicher nicht gewährleistet ist.

Zum anderen ist vom KFC Super aus durch den Befehl MMON der Sprung in den eingebauten Monitor möglich. Der MMON 64.1 kann allerdings auch von Basic aus mit SYS 36864 gestartet werden. Die Befehle des MMON 64.1 sind im wesentlichen mit den Befehlen anderer Monitore vergleichbar.
Folgende Befehle sind möglich:

A	Assemblieren
B	Bits (Sprite Edit)
C	Speicherinhalte vergleichen
D	Disassemblieren
F	Adreßbereich füllen
G	Starten eines Maschinen-Programms
H	Durchsuchen des Adreßbereiches nach vorgegebenen Bytes
L	Laden von Programmen oder Daten
M	Listen der Speicherinhalte
N	Verschieben eines Adreßbereichs
R	Anzeigen der Registerinhalte
S	Abspeichern eines Adreßbereichs
T	Verschiebung eines Adreßbereichs
X	Rücksprung zum Basic
#	Umrechnung Dezimal — Hexadezimal
$	Umrechnung Hexadezimal — Dezimal

Besonders auffallend sind die nicht üblichen Befehle B und C. Mit dem Befehl B wird der Speicherinhalt in einer 3-Byte-Breite bitweise angezeigt. Da die Sprite-Struktur ebenfalls eine 3-Byte-Breite aufweist, können damit, bei entsprechend gesetzten Sprite-Parametern, Sprites editiert werden.

Der Befehl C ermöglicht es, zwei Speicherbereiche miteinander zu vergleichen, was bei Änderungen an bestehenden Programmen sehr hilfreich ist. Insgesammt ist das KFC Super ein interessantes Toolkit, das durch seine Leistungsfähigkeit besticht. Rechnet man die Einzelpreise eines Toolkits, eines Monitors und einer Centronics-Schnittstelle zusammen und vergleicht sie mit dem Preis des KFC Super (198 Mark für den C 64), so steigert sich die Attraktivität des KFC Super zusätzlich.

Zu bedenken, beziehungsweise zu prüfen ist vor dem Kauf die Verträglichkeit mit dem gewünschten Textverarbeitungsprogramm, denn die Aufgabe einer Centronics-Schnittstelle ist es, Texte auf dem Drucker zu ermöglichen. Dies scheint aber beim KFC Super nicht uneingeschränkt möglich zu sein. Auch ist zu beachten, daß ein zusätzliches Kabel gekauft, beziehungsweise angefertigt werden muß, dessen Materialkosten nochmals mit zirka 50 Mark zu Buche schlagen.

Besonders hervorzuheben sind die durchdachten Programmierbefehle, wie zum Beispiel das RENUM, das nicht nur die Zeilennummern, sondern auch Sprungbefehle neu nummeriert, oder der CATALOG Befehl, der keine vorhandenen Programme zerstört. Daß alle Befehle wie im Commodore Basic abgekürzt werden können, erscheint dann schon fast als Selbstverständlichkeit.

Obwohl bei der vorliegenden Testversion die Befehle ORDER zum Anzeigen des Befehlsvorrates und VIEW zum Auflisten der Speicherbelegung und seiner Aufteilung nicht funktionierten, sollte man bei der Auswahl eines Toolkits das KFC Modul mit in Betracht ziehen.

(Arnd Wängler)

Druckfehler-teufelchen

Wir haben uns wirklich bemüht, die erste Ausgabe ohne Fehler zu produzieren. Doch der Teufel steckt im Detail. Wir stehen zu unseren Fehlern und wollen sie nicht totschweigen. Deshalb soll diese Rubrik ein fester — wenn auch möglichst kleiner — Bestandteil des 64'er Magazins werden.

Alle Listings, die im 64'er abgedruckt werden, sind vorher getestet und für gut befunden worden. Die Artikel wurden sorgfältig verfaßt und auf Korrektheit überprüft. Dennoch kann sich in dem einen oder anderen Fall eine Unstimmigkeit oder eine falsche Aussage einschleichen. Wir bitten unsere Leser um Verständnis und eine tätige Mitarbeit. Konstruktive Kritik ist immer willkommen.

Was war nun in der April-Ausgabe nicht richtig, unvollständig oder gar falsch?

SX 64 im Test, Seite 32

Das Bild 4 ist kein kleines Beispiel für die hochauflösende Grafik des SX 64. Das Bild zeigt einen Ausschnitt aus der Christmas-Demo von der Commodore-Weihnachtsdiskette und besteht lediglich aus Grafikzeichen, ist also keine hochauflösende Grafik. Wir danken unserem Leser Detlef Wacker für den Hinweis.

Sprites schneller bewegen, Seite 71

In dem oben genannten Artikel ist folgendes nicht erwähnt worden: Um die Blöcke 32 bis 35 zum Speichern von Sprites zu benutzen, muß der Beginn des Speicherbereichs wie folgt geändert werden.
POKE 44,10:POKE 2560,0: NEW
Beim Abdruck wurde die Änderung der Speicherstelle 44 vergessen

(Herbert Kunz).

Caesar, Seite 78

Die Zeile 1600 lautet korrekt:
1600 GOSUB 10000. In den beiden Zeilen 7450 und 7800 sind die REMs zu entfernen.

Tips & Tricks, Seite 108

Die Joystickabfrage ist für einen korrekten Lauf durch folgende Zeile zu ergänzen:
80 GOTO 20

(Herget Ursula)

Disk Copy, Seite 95

In der letzten Spalte unter »Wichtige Bedienungshinweise«, Punkt 3 muß es richtig heißen:
»PRINT PE« und nicht »PRINT PR«.
Im Listing »Initialisierung« sollte in Zeile 310 der GOSUB-Befehle in: GOSUB 770:... abgeändert werden. Der alte Springbefehl weist auf die REM-Zeile 760. Gibt man das Programm ohne REMs ein, kann der Sprung nicht mehr ausgeführt werden. (Werner Rittmann)
Dieses Programm Disk Copy hat übrigens sehr großen Zuspruch gefunden. Für die Anfänger unter unseren Lesern sei noch erwähnt, daß die Fehlermeldung »BREAK IN 230« nach Ausführung des zweiten Punktes der Bedienungsanleitung völlig korrekt ist und den weiteren Ablauf in keiner Weise beeinflußt.

P.S. Wir suchen noch einen hübschen Namen für unser Fehlerteufelchen. Anregungen aller Art werden gerne entgegengenommen.

P.P.S. Viele Leser haben bei uns angefragt, ob die Hefte 1, 2 und 3 noch zu haben sind.
Die Ausgabe 4 vom April war unser Erstlingswerk. Deshalb können wir auch mit keinen früheren Ausgaben dienen.

(aa)

Die farbig dr

Fast alle heutigen Homecomputer sind in der Lage, Farbe auf den Bildschirm zu zaubern. Nur entsprechende Farbdrucker waren bis vor kurzer Zeit nicht in einer akzeptablen Preislage zu haben. Der zur Zeit wohl billigste Farbdrucker, der in diese Marktlücke gestoßen ist, ist der Seikosha GP-700A Color Printer.

Bild 1. GP-700A ohne Schallschutzhaube. Die Farbbandkassette ist schräg eingebaut.

Der GP-700A ist ein 7-Farb-Matrixdrucker. Der Farbträger (siehe Bild 1 und 2) besteht aus einer Kassette mit einem 4-Farb-Band. Die zusätzlichen drei Mischfarben werden erzeugt durch Übereinanderdrucken. Um die gewünschte Farbe zu erhalten, wird das Farbband in der Höhe verstellt — ähnlich wie man es von den schwarz-roten Farbbändern für Schreibmaschinen her kennt. Der GP-700 kostet rund 1500 Mark. In der Standardausführung wird der Farbdrucker mit einer Centronics-Schnittstelle geliefert (siehe Bild 3). Aber auch Commodore-Besitzer können ihn ohne externes Interface bekommen. Dann wird das C 64-Interface vom Anbieter eingebaut und kostet dann etwas mehr als 1700 Mark. Was bekommt man dafür? Der Bedienungskomfort ist relativ hoch. Der GP-700A besitzt neben der Stop-Taste einen Zeilenvorschubschalter (line feed), einen Seitenvorschub (form feed) und eine Copy-Taste. Diese Taste ermöglicht bei manchen Geräten eine Hardcopy des Bildschirms (low-res), bei anderen Geräten wiederum bewirkt sie nichts, denn diese Funktion ist optional nur gegen Aufpreis erhältlich.

Neben der Darstellung normaler Standardzeichen (Groß- und Kleinschrift mit echten, aber gestauchten Unterlängen) und doppeltbreiten Zeichen ist der GP-700 auch voll grafikfähig. Diese drei Betriebsarten können gleichzeitig auf der gleichen Zeile dargestellt werden. Es ist möglich, den Zeilenabstand sowie die Seitenlänge programmgesteuert zu verändern. Auch die Zeichengröße kann entweder 10 oder 13,3 Zeichen/Zoll betragen. Um im Grafik- und Farbmodus zu arbeiten,

uckende Kreissäge

GRAPHICS CREATED BY PAINTBOX GRAPHIC TOOLKIT FROM PRINT 'N' PLOTTER PRODUCTS

Bild 4. Eine Hardcopy einer hochauflösenden Grafik.

Bild 3. Der Seikosha GP-700A wird standardmäßig mit einer Centronics-Schnittstelle ausgeliefert. Manche Anbieter liefern ihn aber auch mit einer seriellen Schnittstelle für den VC 20 und C 64.

sind eine ganze Reihe von Steuersignalen notwendig, die hier jedoch nicht weiter erwähnt werden sollen.

Wenn man sich die Hardcopys ansieht (siehe Bild 4), die der GP-700 zeichnet, sollte man sich nicht wundern, wenn die dargestellten Farben von den Bildschirmfarben abweichen. Dann wird zum Beispiel ein rotbrauner Farbton orange oder hellblau wird rosa. Aber das sollte nicht weiter stören. Anfangs freut man sich doch über die vielen Farben, und später hat man sich mit diesen Farbverschiebungen abgefunden. Und es ist auch klar: Wenn der Computer 16 Farben darstellt und der Drucker nur in der Lage ist, 7 Farben zu drucken, muß es ja Abweichungen geben.

Der Farbkontrast ist eigentlich relativ hoch — zumindest während der ersten Hardcopys mit einem neuen Farbband. Falls die Tinte zur Neige geht, kann man die vier Tintenbehälter einzeln auswechseln. Was leider nicht gelöst wurde, ist die »Geräuschentwicklung«, die bei der Erstellung einer Hardcopy auftritt.

Bild 2. Die Farbbandkassette des GP-700A. Aus den 4 Grundfarben können noch 3 weitere Mischfarben durch Übereinanderdrucken erzeugt werden.

Aber die hat auch ihre Vorteile: Ein Spaziergang durch das Haus, um sich zum Beispiel die Beine zu vertreten oder bzw. zwei Etagen tiefer in der Küche ein Ei in die Pfanne zu schlagen und es genüßlich zu verspeisen, ist kein Problem: Erstens hört man, wenn der Drucker seine Hardcopy beendet hat und zweitens wird man dann mit dem Essen sowieso schon fertig sein. Nur wenn nebenan jemand mit einer Kreissäge arbeitet, kann man Schwierigkeiten bekommen zu erkennen, was denn jetzt aufgehört hat, die Kreissäge oder der Drucker.

Aber wer sich den Farbdrucker kaufen will, sollte sich überlegen, zu welchem Zweck er die Farbe braucht. Meiner Meinung nach — jeder mag andere Schwerpunkte setzen — kann ein Plotter, was Diagramme und nichtflächige Grafiken betrifft, wesentlich effektiver arbeiten. Nur bei Hardcopys mit großen Farbflächen sehe ich eine möglicherweise sinnvolle Anwendung, etwa zum Einkleben ins Album oder zum Herumzeigen bei (sehr guten) Freunden.

Ein Interessent von Farbdruckern sollte sich überlegen, ob er nicht lieber etwas Geld drauflegt und sich einen entsprechenden Tintenstrahldrucker anlegt. Der ist nämlich auch schon für unter 2000 Mark zu haben. Die farblichen Qualitäten sind zwar nicht viel besser, aber erstens flüstert der Drucker nur und zweitens nimmt die Farbsättigung nicht mit der Zeit ab. (gk)

Der Serielle Bus des VC 20 und des Commodore 64

Über den seriellen Bus werden alle wichtigen Peripheriegeräte, zum Beispiel der Drucker und das Floppy-Laufwerk, an den VC 20 und den Commodore 64 angeschlossen. Im folgenden soll die Arbeitsweise dieses Busses anhand des Betriebssystems des VC 20 unter Berücksichtigung der Eigenheiten beim Commodore 64 vorgestellt werden.

Commodore hat schon bei seinen früheren Modellen PET, CBM 3032 und CBM 8032 eine etwas modifizierte Form des IEEE-488 oder IEC-Bus verwendet. Dieser Bus ist in der Literatur mittlerweile auch für die CBMs sehr gut dokumentiert (siehe Literaturverzeichnis [1] bis [5]). Für den neuen Bus, der beim VC20 und Commodore 64 eingesetzt wird, trifft das leider nicht zu, da er nicht genormt ist und Commodore detaillierte Informationen über ihn nicht oder noch nicht veröffentlicht hat. Der einzige mir bekannte Artikel [8] ist in den USA erschienen und stammt von »Commodore-Guru« Jim Butterfield.

Worin unterscheidet sich nun dieser neue Bus von seinem Vorgänger?

1. Er hat weniger Leitungen
2. Die Daten werden seriell übertragen
3. Er ist langsamer als der IEEE-488-Bus.

Der serielle Commodore-Bus ist physikalisch ein bidirektioneller Bus, der aus sechs Signalleitungen (siehe Tabelle 1) besteht. Der parallele CBM-Bus ist zirka fünfmal schneller. Die Verwandtschaft mit seinem Vorgänger kann er aber nicht verleugnen, denn die Art, wie angeschlossene Geräte angesprochen werden entspricht dem IEEE-Bus. Es gibt zwei Betriebsarten: Den Kommando- und den Datenmodus.

Das gesamte Geschehen auf dem Bus wird von dem »Controller«, unserem VC 20 oder dem C 64 überwacht und gesteuert (Bild 1). Es darf an dem Bus nur ein Controller angeschlossen sein. Alle Geräte dürfen je nach ihren Möglichkeiten als »Talker« Daten auf den Bus geben und als »Listener« Daten vom Bus lesen. Ein Drucker wird nur Listenerfunktionen, ein Floppylaufwerk Listener- und Talkerfunktionen haben. Selbstverständlich besitzt unser Computer beide Funktionen.

Damit die Geräte einzeln vom Computer angesprochen werden können, besitzen sie eine Geräteadresse, die Primäradresse. Sie ist für die Commodoregeräte standardmäßig für den Drucker auf 4 und für das Floppylaufwerk auf 8 gesetzt. Um besondere Befehle an ein Gerät übermitteln zu können, zum Beispiel die Anwahl einer bestimmten Druckart bei einem Drucker, besitzt es häufig noch eine Sekundäradresse. Diese Adressen sind für die Geräte unterschiedlich. Ihre Bedeutung ist dem jeweiligen Handbuch zu entnehmen.

Achtung, ich sende

Wie erfährt nun ein angeschlossenes Gerät, daß es gemeint ist? Nun, dafür gibt es die Leitung »ATN«. Hat der Controller diese Leitung auf »Wahr« (Erklärung siehe weiter unten) gesetzt, unterbricht jedes Gerät seine Tätigkeit, denn es weiß: Jetzt kommt ein Befehl. Die Information, die dann vom Controller auf den Bus gelegt wird, wird von den Geräten gelesen und als Primäradresse interpretiert. Alle Geräte bestätigen den Empfang. Die nicht angesprochenen Geräte setzen, sobald die ATN-Leitung wieder auf »Falsch« gesetzt ist, ihre unterbrochene Tätigkeit fort und kümmern sich nicht mehr um das weitere Geschehen auf dem Bus.

Der Computer teilt dem adressierten Gerät mit, ob es als Talker oder als Listener agieren soll. Über die Sekundäradresse werden vielleicht noch weitere Befehle übermittelt. Dann wird die »ATN«-Leitung vom Computer auf »Falsch« gesetzt. Jetzt kann zwischen Computer und adressiertem Gerät der Datenaustausch stattfinden.

Nachdem das letzte Datum gesendet und empfangen wurde, zieht der Computer die »ATN«-Leitung wieder auf »Wahr«; alle angeschlossenen Geräte reagieren wie oben beschrieben und holen sich wieder die Primäradresse. Der Compter sendet dann zum Beispiel einen »Unlisten« — oder einen »Untalk«-Befehl an das angesprochene Gerät.

Ein Problem gibt es aber noch: Wie weiß der Talker, daß der Listener die Daten auch richtig übernommen hat? Beim IEEE-Bus wird das mit einer Rückmeldung über spezielle Leitungen — den Handshakeleitungen — im Quittungsbetrieb realisiert. Jedes Gerät hat die Möglichkeit, über die jeweilige Leitung den anderen Geräten folgende Informationen zu übermitteln:

DAV die Daten auf dem Datenbus sind gültig (nur Talker)

NDAC die Daten auf dem Datenbus habe ich gelesen (nur Listener)

NRFD ich bin für weitere Daten bereit (nur Listener)

Da aber beim seriellen Bus insgesamt nur sechs Leitungen (siehe Tabelle 1) zur Verfügung stehen, ist der 3-Leitung-Handshake-Betrieb hier icht durchführbar.

Doch bevor wir in den Ablauf des »Handshake-Betriebes« beim seriellen Bus einsteigen, etwas über die »Logik-Pegel« auf dem Bus.

Der Bus wird in negativer Logik betrieben, das heißt ein Low (zirka 0 Volt) wird als »Wahr«, ein High (zirka 5 Volt) als »Falsch« betrachtet.

Es ergibt sich damit folgende Zuordnung zwischen der Spannung auf der Leitung und ihrem logischen Wert:
High = H-Pegel = zirka +5 V = logisch 0 = »Falsch«
Low = L-Pegel = zirka 0 V = logisch 1 = »Wahr«

Low und High

Warum das? Sonst ist es doch in der Digital-Technik genau anders herum! Der Grund liegt darin, daß alle Geräte an den Leitungen angeschlossen sind. Die Ausgänge zum Bus hin sind über »Open-Kollektor-Treiber« realisiert. Die Eingänge sind in normaler Transistor-Transistor-Logik (TTL) ausgeführt. Die Beschaltung einer Leitung des seriellen Busses beim VC 20 ist in Bild 2 am Beispiel der Data-Leitung dargestellt.

Bild 1. Das Bus-System

Bild 2. Die Beschaltung der Busleitungen (die DATA-Leitung beim VC 20)

Signal	Steckerpinnr.	Bedeutung
IFC	6	Interface Clear Leitung um die angeschlossenen Geraete in einen definierten Anfangszustand zu versetzen (RESET)
DATA	5	Leitung fuer Daten (IN/OUT)
CLK	4	CLOCK Leitung um das Handshakeprotokoll der Datenuebertragung (Taktleitung) zu kontrollieren (IN/OUT)
ATN	3	ATTENTION Leitung um mitzuteilen ob Daten oder Kommandos uebertragen werden
GND	2	GROUND Digitale Masse
SRQ	1	SERVICE REQUEST Leitung um dem 'Controller' mitzuteilen, dass Daten bereit stehen (Diese Leitung wird vom Betriebssytem nicht bedient!)

Tabelle 1. Signale und Steckerbelegung des seriellen Busses

Durch diese Schaltungsauslegung ist der aktive Zustand des Busses der L-Pegel, das heißt wenn kein Gerät die Leitung auf 0 Volt zieht, wird die Spannung auf der Busleitung beinahe 5 Volt haben, also High (= Falsch) sein. Ein Bus, an dem keine Peripheriegeräte angeschlossen sind, wird also immer im Zustand »Falsch« verharren — der Buscontroller wird dann beim beim Versuch, ein Gerät anzusprechen, immer die Meldung »Falsch« erhalten und daher einen nicht ordnungsgemäßen Zustand erkennen können.

Noch einen Vorteil hat diese Schaltung. Die Übertragung von Daten darf nicht schneller erfolgen als das langsamste Gerät braucht, um die Daten lesen zu könen. Ein kleines Beispiel:

Ein Talker möchte an zwei Listener Daten senden. Listener 1 ist bereit neue Daten zu übernehmen, er hat also seinen für diese Meldung gedachten Ausgang auf H-Pegel (= »Falsch«) gelegt. Der zweite Listener ist noch nicht soweit, er hält seinen Ausgang noch auf »Wahr«. Damit ist die Leitung, auf die beide Ausgänge arbeiten, ebenfalls »Wahr« — der Talker weiß, daß er noch warten muß. Er legt seinen Ausgang auf »Falsch«. Die Leitung wird, nachdem kein weiteres Gerät sie auf »Wahr« hält, ebenfalls »Falsch« — der Talker weiß, jetzt kann er eine neue Information auf die Datenleitung legen.

Diese Zusammenhänge wollen wir in zwei Merksätzen zusammenfassen:

Wahr wird eine Leitung, wenn mindestens eines der angeschlossenen Geräte »Wahr« sendet.
Falsch wird eine Leitung nur dann, wenn alle Geräte auf dieser Leitung ein »Falsch« senden.

Damit sind die Logikzustände auf dem seriellen Bus beschrieben. Diese Definition ist überaus wichtig, und man muß sie sich ständig bei der Betrachtung des Geschehens auf dem Bus vor Augen halten. Nach diesem kleinen Exkurs in die Welt der Lows und Highs wollen wir jetzt betrachten, wie Daten auf dem seriellen Bus übertragen werden.

Die Datenübertragung

Die für die Datenübertragung wichtigen Leitungen sind die DATA- und die CLK-Leitung. Die Übertragung eines Bytes (= ein Datenwort aus acht Bit) erfolgt bitweise, beginnend mit dem niederwertigsten Bit, das auf die DATA-Leitung gelegt wird. Die Übertragung eines Bytes erfolgt im Handshake-Betrieb, das heißt mit Rückmeldung vom Listener, daß das Byte empfangen wurde und das nächste gesendet werden kann. Verfolgen wir einmal den Ablauf der Übertragung eines Bytes anhand des Zeitablaufdiagrammes Bild 3 und des Ablauf-Diagrammes Bild 4.

Vorbereitung zur Übertragung eines Bytes

CLK- und DATA-Leitung werden beide auf »Wahr« gehalten, wobei der Talker die CLK- und der Listener beziehungsweise alle Listener die DATA-Leitung auf 0 Volt (= »Wahr«) ziehen. Mit diesen Signalen wird jeweils ein »Hier bin ich« signalisiert.

Beginn der Ubertragung

Wenn der Talker ein Byte auf dem Bus ausgeben will, läßt er die CLK-Leitung auf »Falsch« gehen. Dieses Signal entspricht einem »Ich bin bereit, ein Zeichen auszugeben«. Der oder die Listener (in Zukunft werde ich nur mehr von einem Listener sprechen) müssen dieses Signal erkennen und beantworten. Da der Listener zu diesem Zeitpunkt vielleicht noch beschäftigt ist, muß die

```
SENDE BYTE
  Setze CLK-Leitung auf 'Falsch'
  DO UNTIL DATA-Leitung 'Falsch' ist
    Pruefe DATA-Leitung
  EOI ?
  nein:
    Setze CLK-Leitung auf 'Wahr'
  ja:
    Warte mind. 200 us
    DO UNTIL DATA-Leitung 'Wahr' ist
      Pruefe DATA-Leitung
    DO UNTIL DATA-Leitung 'Falsch' ist
      Pruefe DATA-Leitung
    Setze CLK-Leitung auf 'Wahr'
  DO UNTIL alle 8 Bit gesendet sind
    Bit gesetzt ?
    nein: Setze DATA-Leitung auf 'Wahr'
    ja:   Setze DATA-Leitung auf 'Falsch'
    Setze CLK-Leitung auf 'Falsch'
    Warte mind. 20 us
    Setze CLK-Leitung auf 'Wahr'
    Setze DATA-Leitung auf 'Falsch'
  T = 0
  DO UNTIL DATA-Leitung 'Wahr' ist
    Pruefe DATA-Leitung
  T < 1 ms ?
  nein: Fehlermeldung
  ja:   Uebertragung des Byte OK
```

Bild 4. Ablaufschema »Sende ein Byte«

```
                              CLK-Leitung     DATA-Leitung
                              Wahr  Falsch    Wahr  Falsch
Vorbereitung
  Talker   -> CLK  = W
  Listener -> DATA = W
Beginn
  Talker   -> CLK  = F
Bereit fuer Empfang
    ohne EOI =
    mit  EOI = ---
  Listener -> DATA = F        max. 200µs ohne EOI
  Talker   -> CLK  = W
  Listener -> DATA = W        60µs
  Listener -> DATA = F        60µs
  Talker   -> CLK  = W
Uebertragung
  Talker gibt Bit auf
  DATA (W oder F)             max 70µs
  Talker   -> CLK  = F        >20µs
  Talker   -> CLK  = W
  Talker   -> DATA = F
  (nacheinander fuer
  8 Bit wiederholen)          max. 1 ms
Bestaetigung
  Listener -> DATA = W
Zustand wieder wie bei
Beginn (Vorbereitung)
```

Bild 3. Zeitablaufdiagramm »Übertragung eines Bytes« (nicht maßstabgetreu)

```
EMPFANGE BYTE
  DO UNTIL CLK-Leitung 'Falsch' ist
    Pruefe CLK-Leitung
  Setze DATA-Leitung auf 'Falsch'
  T = 0
  DO UNTIL CLK-Leitung 'Falsch' oder T > 200 us
    Pruefe CLK-Leitung
  T > 200 us ?
  nein: keine EOI-Sequenz
  ja:
    Setze DATA-Leitung auf 'Wahr'
    Warte mind. 60 us
    Setze DATA-Leitung auf 'Falsch'
  DO UNTIL 8 Bit empfangen sind
    DO UNTIL CLK-Leitung 'Wahr' ist
      Pruefe CLK-Leitung
    DO UNTIL CLK-Leitung 'Wahr' ist
      Pruefe CLK-Leitung
    Lese Daten-Bit auf DATA-Leitung
    Speichere Bit ab
  Setze DATA-Leitung auf 'Wahr'
```

Bild 5. Ablaufschema »Empfange ein Byte«

Quittierung dieses Signals nicht sofort erfolgen, da hier vom Talker eine Zeitüberschreitung nicht überwacht wird.

Bereit für den Datenempfang

Der Listener ist bereit für das nächste Byte; er gibt die DATA-Leitung frei und sagt damit »Ich bin bereit, Daten zu empfangen«. Wenn alle Listener das getan haben, ist die DATA-Leitung jetzt »Falsch«. Der Talker erkennt das und weiß, jetzt sind alle für neue Daten bereit. Nun gibt es zwei Möglichkeiten, was als nächstes passieren kann:

a. Normales Senden.

Der Talker zieht die CLK-Leitung innerhalb von 200 Mikrosekunden — üblicherweise innerhalb von 60 Mikrosekunden — auf »Wahr«, also auf 0 Volt und bestätigt damit die Meldung. Dann beginnt er das Byte bitweise zu senden.

b. EOI-Meldung (= »Dieses Byte ist das letzte«).

Wenn der Talker die Meldung nicht innerhalb von 200 Mikrosekunden bestätigt, weiß der Listener, daß der Talker ihm mitteilen will, daß das Byte, das jetzt gesendet werden soll, das letzte ist. Es kann das letzte in einem sequentiellen File oder in einem Satz eines relativen Files sein. Wenn der Listener also merkt, daß der Talker das Signal EOI sendet, muß er für mindestens 60 Mikrosekunden die DATA-Leitung auf »Wahr« ziehen und dann wieder auf »Falsch« setzen. Damit hat er dem Talker den Empfang des EOI-Signals bestätigt. Der Talker zieht nun die CLK-Leitung innerhalb von 60 Mikrosekunden auf »Wahr« und beginnt dann das Byte bitweise zu senden.

Die CLK-Leitung ist jetzt, auch wenn die EOI-Sequenz durchgeführt wurde, »Wahr«.

Übertragung des Bytes

Es werden jetzt alle 8 Bits eines Bytes sequentiell, beginnend mit dem niederwertigsten Bit ohne »Handshake« auf die DATA-Leitung gelegt. Beide Leitungen, CLK und DATA werden jetzt allein vom Talker kontrolliert. Für jedes Bit, gesetzt oder nicht, wird die DATA-Leitung auf »Wahr« oder »Falsch« gesetzt. Wenn das Bit auf der DATA-Leitung stabil steht, wird die CLK-Leitung auf »Falsch« gesetzt um mitzuteilen, daß das Bit auf der DATA-Leitung gültig ist. Dieser Vorgang dauert maximal 70 Mikrosekunden. Der Listener hat jetzt wenigstens 20 Mikrosekunden Zeit das Bit zu lesen, da der Talker während dieser Zeit beide Leitungen in diesem stabilen Zustand hält.

Achtung: Für den Commodore 64 muß diese Zeit auf mindestens 60 Mikrosekunden verlängert werden, da ein Interrupt durch den Videocontroller VIC 6567 42 Mikrosekunden dauert und Daten deshalb in dieser Zeit nicht empfangen werden können.

Wie schon oben gesagt, spielt der Listener hier eine komplett passive Rolle. Er wartet darauf, daß die CLK-Leitung auf »Falsch« geht, liest das Bit, speichert es und bereitet sich auf das nächste Bit vor, wenn die CLK-Leitung wieder »Wahr« wird. Nach Ablauf der »Lesezeit« zieht der Talker die CLK-Leitung wieder auf »Wahr«, setzt die DATA-Leitung auf »Falsch« und bereitet sich für das nächste Bit vor.

Empfangsbestätigung

Nachdem das 8. Bit gesendet worden ist, muß der Listener den Empfang des Bytes bestätigen. Da in diesem Moment die CLK-Leitung »Wahr« und die DATA-Leitung »Falsch« ist, zieht der Listener die Data-Leitung auf »Wahr«. Der Talker muß demnach nach dem letzten Bit die Data-Leitung überwachen.

Innerhalb von einer Millisekunde muß sie auf »Wahr« liegen, andernfalls gibt der Talker eventuell eine Fehlermeldung aus, da die Übertragung nicht ordnungsgemäß abgelaufen ist.

Damit sind wir am Ende der Übertragung eines Bytes. Der Talker hält die CLK-, der Listener die DATA-Leitung auf »Wahr«. Wir könnten jetzt beim Punkt »Beginn der Übertragung« das nächste Byte senden, wenn kein EOI aufgetreten ist. Ist ein EOI bei der letzten Übertragung gesendet und empfangen worden, ist die Übertragung beendet, beide Leitungen werden freigegeben und sind damit »Falsch«.

Wir wissen mittlerweile wie ein Byte übertragen wird, doch wie werden zum Beispiel die Primäradresse oder Befehle übertragen? Wie schon weiter oben gesagt, gibt es eine besondere Leitung — ATN — mit der der Controller allen angeschlossenen (und natürlich auch eingeschalteten) Geräten mitteilt »Paßt alle auf! Ich will euch was sagen«. Der Controller ist normalerweise der einzige, der diese Leitung auf »Wahr« ziehen darf. Während er also die ATN-Leitung auf »Wahr« hält, sendet er auf der DATA-Leitung Daten. Die Übertragung dieser Daten erfolgt genauso mit Handshake wie oben beschrieben — nur sind es diesmal eben keine Daten sondern Befehle. Darunter fallen die Busbefehle »Talk«, »Listen«, »Untalk« und »Unlisten«. Auch die Geräteadresse, die Sekundäradresse und die Sekundärbefehle »OPEN« und »CLOSE« werden, während die ATN-Leitung »WAHR« ist, gesendet. Der Empfang dieser Befehle wird von allen angeschlossenen Geräten bestätigt, aber nur die tatsächlich angesprochenen führen den übermittelten Befehl aus. Noch einmal: Wird ATN auf »Wahr« gezogen, unterbrechen alle angeschlossenen Geräte ihre derzeitigen Aktionen. Der Controller zieht die CLK-Leitung auf »Wahr«, da er anschließend Daten senden will.

Die Geräte setzen ihre Ausgänge zur CLK-Leitung auf »Falsch« (sie bleibt aber auf »Wahr«, da der Computer sie auf »Wahr« hält). Genau umgekehrt läuft es bei der DATA-Leitung ab — der Computer läßt sie auf »Falsch« gehen, die Geräte setzen sie auf »Wahr«. Die DATA-Leitung wird vom Computer überwacht und wenn die angeschlossenen Geräte die DATA-Leitung nicht innerhalb von einer Millisekunde auf »Wahr« gezogen haben, wird die Fehlermeldung »DEVICE NOT PRESENT« ausgegeben. Die Übertragung der Daten erfolgt dann auf die schon bekannte Art und Weise.

Ein kleines Beispiel soll zum besseren Verständnis dienen, wie diese Übertragung abläuft. Wir wollen auf dem Gerät 8 — dem Floppylaufwerk — ein Schreibfile eröffnen. Die Basic-Befehlssequenz lautet dann:

OPEN1,8,2, "NAME,S,W"

Der Computer wird die ATN-Leitung auf »Wahr« ziehen, dann die Folge »Gerät 8 Listen, Sekundäradresse 2, Sekundärbefehl OPEN« an das Laufwerk senden. Wenn er dann die ATN-Leitung wieder auf »Falsch« setzt, wartet er als Talker mit der CLK-Leitung auf »Wahr«. Das Laufwerk hält als Listener die DATA-Leitung auf wahr. Hiermit ist der Zustand »Vorbereitung zur Übertragung eines Bytes« (siehe oben) hergestellt. So muß es ja auch sein, da der Computer noch weitere Informationen senden will. Er wird als nächstes dann byteweise den Namen, Komma, »S«, Komma und »W« senden. Das »W« wird, da es das letzte Byte ist, mit der »EOI«-Sequenz übermittelt. Danach wird der Computer mit der ATN-Leitung wieder auf »Wahr« ein »Gerät 8 Unlisten«-Befehle senden. Damit ist das File als Schreibfile eröffnet.

Computer als Zuhörer

Irgendwann will dann der Computer Daten auf dieses File schreiben. Er sendet also wieder mit der ATN-Leitung auf »Wahr« ein »Gerät 8 Listen« gefolgt von »Sekundäradresse 2«. Er setzt die ATN-Leitung auf »Falsch« und beginnt dann, wie oben beschrieben, die Daten an das Laufwerk zu senden. Nur das Floppylaufwerk wird diese Daten empfangen und sie in das File mit dem Namen »NAME« schreiben. Das letzte Byte, das der Computer sendet, wird wieder mit »EOI« gesendet. Mit der ATN-Leitung wieder auf »Wahr« gesetzt, sendet er einen »Gerät8Unlisten«-Befehl. Diese Abfolge kann im Laufe einer Programmabarbeitung öfters durchlaufen werden. Zu irgendeinem Zeitpunkt soll dieses Schreibfile wieder mit dem Basic-Befehl

CLOSE 1

geschlossen werden. Der Computer wird also, natürlich mit der ATN-Leitung auf »Wahr«, die Folge »Gerät 8 Listen, Sekundäradresse 2, Sekundärbefehl CLOSE« an das Laufwerk senden. Die 1 bei CLOSE dient im Computer nur als Zeiger auf die tatsächlichen OPEN- und CLOSE-Parameter. Über diese 1 findet der Computer dann die Sekundäradresse 2.

Bis jetzt haben wir den Computer eigentlich immer in der Funktion als Controller und als Talker betrachtet. Wie wir wissen, bleibt er immer der Buscontroller, aber er kann auch als Listener fungieren. Diesen Ablauf haben wir bis jetzt noch nicht beschrieben.

Im Prinzip kennen wir schon alle dazu nötigen Schritte. Während die ATN-Leitung auf »Wahr« ist, sendnet der Computer einen Talk-Befehl an das adressierte Gerät. Unmittelbar nachdem die ATN-Leitung wieder auf »Falsch« geht, ist das Gerät immer noch Listener und der Computer Talker. Die DATA-Leitung wird vom Gerät, die CLK-Leitung vom Computer auf »Wahr« gehalten. Das muß sich jetzt aber ändern.

Der Computer zieht seinen DATA-Ausgang auf »Wahr« (die Leitung ist es schon durch das Gerät) und setzt seinen CLK-Ausgang auf »Falsch«. Darauf hat das Peripheriegerät gewartet. Es gibt seinen DATA-Ausgang frei (»Falsch«) und zieht die CLK-Leitung auf »Wahr«. Es hat sich also jetzt wieder der Zustand "Talker hält die CLK-, Listener die DATA-Leitung auf »Wahr«" (siehe den Punkt »Vorbereitung zur Übertragung eines Bytes«) eingestellt. Dieser Ablauf wird vom Computer überwacht und nur wenn alles korrekt abläuft, ist er bereit, Daten zu empfangen. Die Übertragung der Daten erfolgt auf die gleiche Weise wie oben beschrieben.

Damit haben wir alle Übertragungsmöglichkeiten auf dem seriellen Bus erläutert. Wer es verstanden hat, kann die beiden folgenden Fragen beantworten: Warum gibt der Computer keine Fehlermeldung beim Basic-Befehl OPEN4,4 aus, obwohl nur das Gerät 8 angeschlossen und in Betrieb ist (das Gerät 4 ist nicht angeschlossen)? Warum gibt er aber eine Fehlermeldung bei PRINT@4, "xxxx" aus (' steht für das Anführungszeichen)?

(Michael Bauer)

Literaturverzeichnis:
1) IEC-Bus, Grundlagen — Technik — Anwendungen; Elektronik Sonderheft Nr. 47; Franzis Verlag; München
2) Geräte mit dem IEC-Bus einfach gesteuert; Computer Persönlich Nr. 25 vom 15.12.1982 Seite 76; Verlag Markt und Technik; Haar
3) Der IEC-Bus-Standard; Computer Persönlich Nr. 24 vom 17.11.1983 Seite 97; Verlag Markt und Technik; Haar
4) Piotrowski, Dr. Anton; IEC-Bus; Franzis Verlag; München
5) Wunderlich, Franz und Geissen, Bernd; IEE-Bus, Userport + VO-Register; Info-Dienst 1-4; Commodore Benutzer Klub; Frankfurt
6) VC 20 Programmierhandbuch; Commodore; Frankfurt
7) C 64 Programmierhandbuch; Commodore; Frankfurt
8) Butterfield, Jim; How the VIC/64 Serial Bus Works; Compute! Juli 1983 Seite 178; Greensboro
9) Englisch, Szczepanowski; Das große Floppy-Buch; Data Becker; Düsseldorf

STRUKTURIERTES PROGRAMMIEREN

Strukturiertes Programmieren ist keineswegs so schwierig wie es sich anhört. Neben etwas Theorie wollen wir ein kleines aber ausbaufähiges Spiel entwerfen.

Im letzten Heft gab ich Ihnen einige Regeln, die das Strukturieren des Programmcodes betreffen. Eigentlich war dieser Bericht zu früh dran. Denn bevor man sich an seinen Computer setzt und das Programm eintippt, ist eine Menge Vorarbeit zu leisten. Die wichtigste Arbeit besteht dabei im Nachdenken. Nachdenken vor allem darüber, was das Programm, das man schreiben will, eigentlich tun soll, wie man es am effektivsten aufbaut. Und dabei werden die meisten und verhängnisvollsten Fehler gemacht. Diese Fehler ziehen sich durch das gesamte »Software-Projekt« hindurch und entscheiden später über den Erfolg oder Mißerfolg des Programms. Und je später ein Fehler endeckt wird, desto größer wird der Aufwand, den man betreiben muß, um diesen Fehler zu beheben. Das bedeutet ganz einfach, daß man am Anfang sehr aufmerksam sein muß, und jeden Schritt, jede Idee lieber einmal mehr als einmal zu wenig überdenken sollte.

Gerade dann, wenn das Programm wahrscheinlich etwas größer wird, tut man gut daran, es in kleinere Einheiten aufzuteilen. Diese Einheiten werden auch Module genannt. Gebräuchlich sind auch die Begriffe »Unterprogramm«, »Subroutine« oder auch »Prozeduren« (zum Beispiel in Pascal). Gemeint ist in der Regel immer das gleiche.

Bei großen und komplexen Programmpaketen besteht eine der wichtigsten Aufgaben in der Modularisierung eines Programms. Das heißt aber auch, daß, wenn diese Aufgabe erste einmal gelöst ist, der Rest um so einfacher und schneller von der Hand geht.

Wie ein Spiel entsteht

Wenden wir uns jetzt einem neuen Beispiel zu. Ich möchte mit Ihnen ein kleines und einfaches Spiel entwerfen. Die Spielregeln sind sehr einfach. Wir wollen den Computer veranlassen, gegen uns zu spielen. In ein Spielfeld von 3 x 3 Feldern setzen die Spieler (der Computer und wir) abwechselnd eine Null oder ein X. Wir setzen die X. Wer als erster drei Zeichen in einer Reihe hat, hat gewonnen. Eine Reihe heißt hier senkrecht, waagerecht oder diagonal. Das Spielfeld und eine typische Spielsituation sind in Bild 1a und 1b zu sehen.

x	0	
x		x
0		

1	2	3
4	5	6
7	8	9

Bild 1a, b, c. Das Spielfeld wird eine typische Spielsituation

Wir wollen unser Spielfeld jetzt numerieren, jedes Feld erhält eine Nummer (Bild 1c).

Jetzt schreiben wir uns auf, was in jedem Feld steht:

```
1 x   2 0   3 —
4 x   5 —   6 x
7 0   8 —   9 —
```

Ein Strich bedeutet hier ein leeres Feld.

```
Ersetze x durch 1
        0 durch —1
        — durch 0
```

Der Grund dafür, daß wir gerade diese Zahlen nehmen, ist jetzt noch nicht so klar, im Moment ist es einfach ein Gefühl für Symmetrie. Wir werden aber sehen, daß diese Wahl zweckmäßig ist.

Damit können wir den Spielstand so beschreiben:

Feld	Inhalt
1	1
2	—1
3	0
4	1
5	0
6	1
7	—1
8	0
9	0

Array SS (Spielstand)

Diesen (und auch jeden anderen) Spielstand legen wir in einem Array SS ab (SS steht für Spielstand). So oft sich also der Spielstand ändert, ändert sich auch der Inhalt des Arrays SS. SS(5) enthält also immer den Inhalt des mittleren Feldes, eine 1, wenn wir einen Stein (x) auf das Feld gesetzt haben, eine —1, wenn der Computer seinen Spielstein darauf gesetzt hat, oder eine 0, wenn das Feld noch nicht besetzt ist.

Jetzt können wir (und vor allem der Computer) herausfinden, welches Feld besetzt ist, und auch, durch wen es besetzt wurde, aber wir müssen den Compuer ja noch dazu bringen, einen vernünftigen Spielzug zu machen. Das bedeutet, er muß sperren, wenn wir schon zwei Felder einer Reihe besetzt haben, und er soll auch erkennen, wenn er schon zwei Felder einer Reihe besetzt hat, wo er seinen Siegzug hinzusetzen hat. Wir brauchen also eine Methode, um den Spielstand zu bewerten.

Dazu schaffen wir ein zweites Array, das wir BW nennen (BW steht für Bewertung). BW enthält die Summen jeder Reihe, der waagerechten, der senkrechten und der diagonalen. Insgesamt haben wir acht Reihen, so daß auch BW nicht größer zu sein braucht. Und so wird jedes Element von BW berechnet:

1	0	=	SS(1) + SS(2) + SS(3)	oberste Reihe
2	2	=	SS(4) + SS(5) + SS(6)	zweite Reihe
3	—1	=	SS(7) + SS(8) + SS(9)	dritte Reihe
4	1	=	SS(1) + SS(4) + SS(7)	linke Spalte
5	—1	=	SS(2) + SS(5) + SS(8)	mittlere Spalte
6	1	=	SS(3) + SS(6) + SS(9)	rechte Spalte
7	1	=	SS(1) + SS(5) + SS(9)	vordere Diagonale
8	—1	=	SS(3) + SS(5) + SS(7)	hintere Diagonale

Diese Werte geben selbstverständlich nur den oben im Beispiel vorgegebenen Spielstand wieder. Aber wir können jetzt erkennen, daß das Array BW uns Informationen über die aktuelle Spielsituation liefert: Immer, wenn ein Element von BW den Wert 2 hat, erkennt der Computer, daß für ihn Gefahr im Verzuge ist. Die 2 bedeutet nämlich, daß in einer Reihe schon zweimal ein Sein gesetzt wurde. Damit dürfte die nächste Aufgabe des Computers schon klar sein: Er muß seinen nächsten Stein in das dritte Feld der fast kompletten Reihe setzen! Doch wie soll er das freie Feld finden? Wir machen es uns einfach:

Er soll so lange einen Stein in ein freies Feld setzen, bis eine erneute Überprüfung des Arrays BW ergibt, daß kein Element mehr den Wert 2 hat. Und wirklich, wenn der Computer zufällig das richtige Feld erwischt hat, wird ja dieses Feld mit einer —1 belegt. Dadurch wird in dem BW-Feld, in dem eine 2 steht, —1 hinzuaddiert, also 1 abgezogen. Damit existiert in diesem Moment keine gefährliche Situation mehr, jedenfalls nicht für den Computer. Ich sagte gerade, daß der Computer ganz zufällig irgendein freies Feld belegt, um dann nachzuprüfen, ob sich dadurch eine Entschärfung der für ihn gefährlichen Situation ergibt. Falls er ein Feld belegt hat, das nicht zu einer Lösung führt (das heißt die 2 im Feld BW wird nicht verändert), müssen wir natürlich dafür sogen, daß diser Zug wieder zurückgenommen wird. In diesem Fall braucht also dieses Feld nur wieder mit einer 0 belegt zu werden. Aber warum meinte ich »zufällig«? Nun, wir könnten das Spielfeld systematisch druchsuchen lassen, aber dann wäre jeder Zug vorhersehbar und bei einem Spiel wäre das nicht besonders interessant.

Die Funktionen

Wir legen jetzt also fest, daß der Computer nur dann zu sperren braucht, wenn sich eine für ihn gefährliche Situation ergibt, wenn also zwei Felder in einer Reihe durch ein X besetzt sind. Alle anderen Fälle interessieren ihn nicht besonders, in diesen Fällen setzt er seine Steine ganz zufällig auf ein freies Feld. Selbstverständlich bemerkt er auch, wenn er selbst zwei Felder besetzt hat. Dann versucht er natürlich, daß dritte Feld auch zu belegen. Und last not least, erkennen soll der Computer auch, wenn er oder wir gewonnen haben. Und das erkennt er dann, wenn die Summe einer Reihe +/—3 ergibt.

So, mit diesen Ideen können wir schon etwas anfangen. Wir versuchen jetzt, die einzelnen Funktionen des Programms festzulegen. In welche Teilaufgaben können wir das Problem zerlegen? Als erstes brauchen wir eine Initialisierung der Variablen. Das Spielfeld soll auch angezeigt werden, das ist ein weiterer Teil. Dann gibt es einmal den Computerzug und auch unseren Spielzug. Und auch die Überprüfung auf Spielende darf nicht fehlen.

Daraus ergibt sich:

Funktion	Beginn einer Zeile
Hauptprogramm	10
Initialisierung des Spielfeldes	1000
Anzeigen des Spielfeldes	2000
Zug holen	3000
Auf Spielende prüfen	4000
Computerzug	5000

1000 Initialisierung → 2000 Anzeigen Spielfeld → 3000 Zug holen → 2000 Anzeigen Spielfeld → 4000 Spielende prüfen → 5000 Computerzug → 2000 Anzeige Spielfeld → 4000 Spielende prüfen (→ zurück zu 3000)

Bild 2. Das Ablaufdiagramm

In der Reihenfolge wie sie Bild 2 zeigt, werden die einzelnen Module aufgerufen. Deshalb nennt man dies auch ein Ablaufdiagramm. Manche dieser Module müssen noch weiter unterteilt werden. Zum Beispiel der Modul SPIELENDE (4000): Um ein Spielende zu erreichen, gibt es mindestens zwei Möglichkeiten: Entweder ist eine Reihe mit drei gleichen Steinen besetzt oder aber das Spielfeld ist voll. Dann gibt es ein Unentschieden.

Um das herauszufinden, erstellen wir eine Routine zum BEWERTEN (6000) des Spielfeldes. In diesem Modul wird das Array BW erzeugt (siehe oben).

Wie steht es nun mit dem Modul COMPUTERZUG (5000)?

Ganz klar, auch der Computer muß wissen, wie das Spielfeld jetzt aussieht. Deswegen rufen wir auch hier den Modul BEWERTEN auf. Aus dieser Bewertung heraus können sich drei Möglichkeiten ergeben:

1. Der Computer hat schon eine Reihe mit zwei Steinen besetzt und das dritte Feld ist noch frei (das heißt ein Element von BW hat den Wert —2). Dann führt er einen SIEGZUG (7000) aus. Da wir schon vorhin davon sprachen, daß der Computer seinen Stein zufällig setzt, müssen wir hier den Modul ZURÜCKNEHMEN (10000) berücksichtigen

2. Die Gegner — also wir — haben schon zwei Steine in einer Reihe. Wenn das dritte Feld dieser Reihe noch frei ist, ärgert uns der Computer, indem er dieses Feld versucht zu SPERREN (8000).

3. Im dritten Fall ergibt sich weder das eine noch das andere, und der Computer macht einen ZUFALLSZUG (9000).

Selbst wenn wir die Frechheit besitzen, den Computer auszutricksen, in dem wir zwei Reihen mit je zwei Steinen und einem freien Feld erspielen können, merkt er das und verabschiedet sich nachtragend.

Damit wäre auch diese Ebene definiert.

Unser Programm sieht jetzt so aus:

0	HAUPTPROGRAMM	10
1.1.	INITIALISIERUNG	1000
1.2.	ANZEIGEN	2000
1.3.	ZUG HOLEN	3000
1.4.	SPIELENDE	4000
1.5.	COMPUTERZUG	5000
2.1.	BEWERTEN	6000
2.2.	SIEGZUG	7000
2.3.	SPERREN	8000
2.4.	ZUFALLSZUG	9000
3.1.	ZURÜCKNEHMEN	10000

Jetzt haben wir alle notwendigen Module benannt. Daraus erstellen wir ein Übersichtsprogramm. In diesem Übersichtsprogramm werden alle Module grafisch dargestellt und zwar in ihren Abhängigkeiten voneinander (siehe Bild 3).

0 — 10 Hauptprogramm
1.1 — 1000 Initialisierung
1.2 — 2000 Anzeige Spielfeld
1.5 — 5000 Computerzug
1.3 — 3000 Zug holen
1.4 — 4000 Spielende prüfen
2.4 — 9000 Zufallszug
2.3 — 8000 Sperren
2.1 — 6000 Spielfeld bewerten
2.2 — 7000 Siegzug
2.1 — 6000 Bewerten
2.4 — 9000 Zufallszug
2.1 — 6000 Bewerten
3.1 — 10000 Zurücknehmen
2.4 — 9000 Zufallszug
2.1 — 6000 Spielfeld bewerten
3.1 — 10000 Zurücknehmen

Bild 3. Das Übersichtsdiagramm

Die Beziehung zwischen den einzelnen Unterprogrammen ist: »ruft auf«. Das Hauptprogramm also ruft auf: INITIALISIERUNG, ANZEIGEN, ZUG HOLEN, SPIELENDE PRÜFEN und COMPUTERZUG. Manche der anderen Module werden nur nach einer bedingten Verzweigung (IF ... THEN GOSUB ...) durchlaufen. Das spielt bei dieser Übersicht jedoch keine Rolle.

So, jetzt sind wir so weit, daß wir uns die einzelnen Teile etwas genauer anschauen können. Fangen wir wieder mit dem Hauptprogramm an. Oben im Ablaufdiagramm ist es eigentlich schon vollständig beschrieben. Hier könnten wir schon sofort das Coding hinschreiben (Listing 1).

Wenn Sie dieses Listing mit dem Ablaufdiagramm (Bild 2) vergleichen, sehen Sie die direkte Umsetzung unserer Überlegungen. Lediglich der Programmkopf, das Löschen des Bildschirms (Zeile 90) und die Dimensionierung unserer Arrays (Zeile 100) kommt noch hinzu. Diese Zeile ist in unserem Beispiel zwar nicht notwendig (der C 64 läßt Arrays bis 11 Elemente zu ohne sie dimensionieren zu müssen), aber da wir ja weiterdenken, bereiten wir unser Programm schon für spätere Erweiterungen vor (es könnte ja sein, daß unser Spielfeld vergrößert werden soll). Außerdem erkennen wir schon im Hauptspeicher, welche Variablen für das ganze Programm benötigt werden. Man spricht hier auch von »globalen Variablen«, im Gegensatz zu »lokalen Variablen«, die nur innerhalb eines Moduls wichtig sind.

Jetzt, nachdem unser Hauptprogramm »steht«, knöpfen wir uns das nächste Modul vor.

INITIALISIERUNG

In diesem Modul wollen wir die Felder unseres Spielfeldes löschen. Das bedeutet, wir löschen Array SS (Listing 2).

```
10 REM ********************************
20 REM *  KREUZ UND QUER              *
30 REM ********************************
90 PRINT"♥"
100 DIMSS(9):DIMBW(8)
110 GOSUB1000:REM SPIELFELD INITIALISIEREN
120 GOSUB2000:REM ANZEIGEN
130 GOSUB3000:REM ZUG HOLEN
140 GOSUB2000:REM ANZEIGEN
150 GOSUB4000:REM AUF SPIELENDE PRUEFEN
160 GOSUB5000:REM COMPUTERZUG
170 GOSUB2000:REM ANZEIGEN
180 GOSUB4000:REM AUF SPIELENDE PRUEFEN
190 GOTO130
READY.
```

Listing 1. Das Hauptprogramm unseres Beispiels

```
1000 REM---------------
1010 REM INITIALISIERUNG
1020 :
1100 FORP=1TO9
1110 :SS(P)=0
1120 NEXTP
1130 RETURN
READY.
```

Listing 2. Initialisierung des Feldes ss

Ich will für einige der folgenden Module das Struktprogramm erstellen. Die Umsetzung in ein ablauffähiges Programm überlasse ich Ihnen. Eine Ausnahme werde ich

Anzeigen (2000)

FOR P = 1 bis 9

SS(P) = 1
N J
PRINT "X";

SS(P) = —1
N J
PRINT "O";

SS(P) = 0
N J
PRINT ".";

INT(P/3) = P/3
N J
PRINT

NEXT P

RETURN

Bild 4. Nassi-Schneidermann-Diagramm: Spielfeldanzeigen

SPIELERZUG (3000)

Eingabe
INPUT "ZUG EINGEBEN";P

Feld besetzt?
SS(P)< >0
N Ja
Ausgabe:
"Feld besetzt"
GOTO Anfang

Feld besetzen
SS(P) = 1

RETURN

Bild 5. Modul Spielzeug

machen mit den Modulen COMPUTERZUG, SPIELFELD BEWERTEN und ZURÜCKNEHMEN.

Modul SPIELFELD ANZEIGEN (2000/) (Bild 4)

Modul SPIELERZUG (3000/) (Bild 5)

Modul SPIELENDE PRÜFEN (4000/) (Bild 6)

Modul COMPUTERZUG (5000/) (Bild 7)

Und dann der versprochene Code dazu: (Listing 3)

Modul SPIELFELD BEWERTEN (6000)

AUF SPIELENDE PRÜFEN (4000)

BEWERTEN 6000

FOR P = 1 TO 9

BW(P) = —3
N Ja
AUSGABE:
"Ich gewinne"
END

BW(P) = +3
N Ja
Ausgabe:
"DU
GEWINNST"
ENDE

NEXT P

FOR P = 1 TO 9

SS(P) = 0
N Ja
RETURN

Next P

Ausgabe "UNENTSCHIEDEN"

END

Bild 6. Auf Spielende prüfen

Hier wird das Array BW definiert. Man könnte dieses Modul natürlich mit einer Schleife lösen, aber eigentlich lohnt sich der Aufwand nicht (Listing 4).

Der Modul SIEGZUG (7000) ist auch schnell gelöst (Bild 8).

Modul SPERREN (8000/) (Bild 9)

Modul ZUFALLSZUG (9000/) (Bild 10)

Modul ZURÜCKNEHMEN (100000)

Und dies ist die einfachste Routine, ein Zweizeiler:

```
10000 REM ——————————————
10010 REM ZURÜCKNEHMEN
10030 :
10040 SS(CM) = 0
10050 RETURN
```

So, daß war das ganze Programm! Eigentlich sollte es Ihnen gelingen, anhand der Struktogramme das Programm zu schreiben. Aber Sie sehen schon jetzt, daß eine Änderung des Programms ganz einfach ist. Man kann sich jeden Modul einzeln vornehmen und ihn

COMPUTERZUG (5000)

BEWERTEN 6000

MT = 0 (Anzahl Reihen mit gefährlicher Situation)

FOR P = 1 bis 8

SIEGZUG möglich?
BW(P) = —2
Ja
SIEGZUG 7000
RETURN

Gegner Gewinnchance
BW(P) = 2
Ja
MT = MT+1

Next P

Mt=2
N Ja
Ausgabe
"Auch gut, Angeber!"
END

Mt = 1
N Ja
SPERREN 8000
RETURN

Zufallszug 9000

RETURN

Bild 7. Computerzug

SIEGZUG (7000)

ZUFALLSZUG 9000

Bewerten 6000

FOR P = 1 TO 8

BW(P) = —3
Ja
RETURN

NEXT P

ZURÜCKNEHMEN 10000

GOTO Anfang

Bild 8. Modul Spielzug

je nach Wunsch ändern oder erweitern. Dazu bietet sich zum Beispiel der Modul SPIELFELD ANZEIGEN (2000) an (Bild 4). Man kann die Bildschirmdarstellung sehr gut verbessern, ohne die Programmstruktur zu ändern. Man könnte auch die Strategie des Com-

```
5000 REM----------------------------
5010 REM COMPUTERZUG
5020 :
5030 GOSUB6000
5040 MT=0
5050 FORP=1TO8
5060 :IFBW(P)=-2THENGOSUB7000:RETURN
5070 :IFBW(P)=2THENMT=MT+1
5080 NEXTP
5090 IFMT=2THENPRINT"AUCH GUT. ANGEBER!":END
5100 IFMT=1THENGOSUB8000:RETURN
5110 GOSUB9000
5120 RETURN
5130 :
```

Listing 3. Computerzug

```
6000 REM-------------------
6010 REM SPIELFELD BEWERTEN
6020 :
6030 :
6040 BW(1)=SS(1)+SS(2)+SS(3)
6050 BW(2)=SS(4)+SS(5)+SS(6)
6060 BW(3)=SS(7)+SS(8)+SS(9)
6070 BW(4)=SS(1)+SS(4)+SS(7)
6080 BW(5)=SS(2)+SS(5)+SS(8)
6090 BW(6)=SS(3)+SS(6)+SS(9)
6100 BW(7)=SS(1)+SS(5)+SS(9)
6110 BW(8)=SS(3)+SS(5)+SS(7)
6120 RETURN
6130 :

READY.
```

Listing 4. Spielfeld bewerten

puters im Modul COMPUTERZUG (5000) verbessern (Bild 7), zum Beispiel soll er zuerst das Feld 5 besetzen (wenn es noch frei ist). Das bringt Spielvorteile (vorausgesetzt, es besteht keine Bedrohung oder eine Gewinnposition ...). Auch könnte man abwechselnd den Computer oder den Mitspieler anfangen lassen zu spielen. Ändern Sie das Programm so ab, daß auch mehrere Spiele durchgeführt werden können und das darüber Buch geführt wird. Sie werden feststellen, daß Änderungen durch die Modularisierung relativ einfach gemacht werden können. Erkennen Sie den Vorteil dieser Programmstrukturierung?

Bild 9. Modul Sperren

Bild 10. Zufallszug

Was wir hier gemacht haben, ist ein Top-down-Entwurf. Wir haben bei dem Modul an der Spitze der Hierarchie angefangen (beim Steuermodul oder Hauptprogramm) und entwickelten dann die weiteren Module. Es gibt noch eine andere Methode. Sie nennt man Bottom-up-Methode. Bei dieser Methode ist es genau umgekehrt: Hier fängt man an der untersten Hierarchiestufe an und endet an der obersten.

Im nächsten Heft stelle ich Ihnen die Elemente der Flußdiagramme und auch die Struktogramme ausführlich vor. Und wenn Sie Lust haben, schicken Sie uns doch Ihre Lösungsvorschläge des Programms. Ich könnte mir vorstellen, daß wir einige interessante Lösungen erhalten werden.

(Dieses Beispiel ist zum Teil entnommen aus dem Buch Commodore 64, Programmieren leicht gemacht, aus dem Birkhäuser Verlag (siehe auch unsere Buchbesprechung)).

TOP-DOWN-Vorgehensweise

Top-Down ist eine bestimmte Vorgehensweise bei der Entwicklung und beim Test von Programmen. Sie basiert auf hierarchischen Strukturen und dient als sinnvolle Ergänzung zur strukturierten Programmierung. Zuerst werden die im Sinne der Hierarchie obersten Programmteile entworfen, codiert und getestet, ehe zu Programmteilen der nächst niedrigeren Hierarchiestufe übergegangen wird. Dadurch sind Programmierobjekte besser überschaubar, allgemeine Anforderungen können zuerst, Details später hinzugefügt werden. Weil man sich bei diesem Vorgehen am Anfang mehr Mühe machen muß, wird durch die Betrachtung des Gesamtsystems das frühzeitige Erkennen von Entwurfsfehlern erleichtert. Der Einbau von Entwurfsänderungen ist in den meisten Fällen problemlos.

BOTTOM-UP

Im Gegensatz zum TOP-DOWN-Entwurf fängt man in einer hierarchischen Programmstruktur auf der untersten Stufe an, entwickelt die einzelnen Komponenten, testet sie einzeln. Ist der Test erfolgreich verlaufen, geht man zur nächsten, höheren Hierarchiestufe weiter und setzt die Arbeit ähnlich fort. So wird ein Programm stufenweise von unten nach oben entwickelt und getestet. Diese Vorgehensweise hat zweifellos Vorteile, man darf jedoch ihre Nachteile nicht übersehen:

Je höher die Integrationsstufe, also je höher man in der Hierarchie gestiegen ist, um so schwieriger ist die Ursache von Fehlern zu entdecken.

Änderungen »in der letzten Minute« verursachen wiederum neue Fehler und verschlechtern die Qualität gut konzipierter Programme.

Zusammenhängende Ergebnisse werden während der ganzen Entwicklungsphase nicht sichtbar.

Wie bitte, Sie besitzen Programm und wissen es gar nicht?

Mit jedem Diskettenlaufwerk 1541 wird eine Demo-Diskette mitgeliefert. Auf dieser Diskette sind einige Programme, von denen der von jeglichem Computerwissen unberührte Neuling nicht genau weiß, wie er sie einzusetzen hat. Unter anderem ist das das Programm DOS 5.1. Dieses Programm erlaubt die bequeme Nutzung vieler Diskettenbefehle.

Was macht das DOS 5.1?

Es liefert drei verschiedene Arten von Befehlen:
— Der Fehler-Status kann mit einem einfachen Befehl gelesen werden;
— das Inhaltsverzeichnis (Directory) einer beliebigen Diskette kann geladen werden, ohne ein im Computer befindliches Programm zu zerströren;
— eine Anzahl von Diskettenbefehlen kann einfach durchgeführt werden.

Wie man das DOS 5.1 in den Computer lädt

Vielleicht haben Sie es schon einmal mit LOAD"DOS 5.1",8 probiert und haben dann lediglich ein Syntax-Error erhalten. Das liegt daran, daß das DOS ein Maschinenprogramm ist. Es gibt auf der Diskette aber ein anderes Programm, das sich »C-64 Wedge« nennt. Und dieses Programm ist das Ladeprogramm für das DOS 5.1. Also laden Sie es mit LOAD"C-64 WEDGE",8 und drücken dann RUN. Dadurch wird das DOS automatisch in den Speicher des Computers geladen. Nach kurzer Zeit meldet sich der Computer mit einer kurzen Meldung (siehe Bild). Jetzt können Sie

Ich bin sicher, daß ein großer Teil aller C-64-Besitzer, die sich eine Floppy zugelegt haben, gar nicht wissen, was sie für ein nützliches und interessantes Programm in ihrem Besitz haben. Oder sie wissen damit nicht viel anzufangen. Gemeint ist das DOS 5.1 auf Ihrer Demo-Diskette.

das Programm nutzen. Es macht nichts, wenn Sie jetzt »NEW« eintippen: Das DOS 5.1 liegt in einem Speicherbereich, der nicht mit »NEW« gelöscht werden kann. Sie können auch ruhig irgendein anderes Basic-Programm laden. Es ist ohne weiteres möglich, und die beiden Programme beeinflussen einander nicht.

Die Befehle des DOS 5.1

DOS-Befehle müssen als direkte Befehle eingegeben werden, das heißt, man kann sie nicht innerhalb von Programmen einsetzen. Die meisten DOS-Befehle beginnen mit einem »>« oder »@«. Man kann sie wahlweise benutzen, sie bewirken das gleiche. Da jedoch das »@« mit einer Taste gedrückt werden kann, werde ich im folgenden nur dieses Zeichen verwenden. Doch dazu später. Es gibt noch andere Zeichen, mit denen DOS-Befehle beginnen, das sind die Zeichen (←), (↑) und (/). Diese Zeichen haben folgende Bedeutung:

(←): Speichern (Save) eines Programms.
Beispiel: ←TEST speichert ein Programm namens TEST auf die Diskette. Sonst würde man schreiben SAVE"TEST", 8
↑ lädt ein Programm (LOAD) von Diskette und startet es automatisch.
Beispiel: ↑TEST lädt das Programm TEST von der Diskette und startet es automatisch und ersetzt die Befehle LOAD"TEST",8:RUN.
/ lädt ein Programm, ohne es zu starten. Beispiel: /Test entspricht dem LOAD"TEST',8

In vielen Fällen ist es nicht nötig, beim Laden und Speichern eines Programms den Programmnamen voll auszuschreiben. Man kann Teile des Namens eingeben und »?« beziehungsweise »*« für den Rest eintippen.

Beispiel: Der Befehle /TE* lädt das erste Programm von der Diskette, dessen erste beiden Buchstaben Te sind, das kann das Programm TE sein, aber auch TEST oder TERMIN oder TEUER, falls diese Programme auf der Diskette vorhanden sind. Der Befehl /* lädt das erste Programm auf der Diskette.

Das »?« ersetzt irgendein Zeichen im Programmnamen: /T??T lädt jedes Programm, dessen erster und vierter Buchstabe ein T ist. Das könnte ein Programm namens TEST, TANT, TORT, und so weiter sein.

Benutzen Sie »?« und »*« aber nicht für den SAVE-Befehle (SAVE beziehungsweise ←). Beim Abspeichern muß der Programmname natürlich voll ausgeschrieben werden!

Disk-Status

Dies ist der kürzeste Befehl von DOS 5.1. Sie brauchen lediglich das @-Symbol einzutippen, gefolgt von

ein tolles

der Return-Taste, und Sie erhalten den Floppy-Status angezeigt. Wenn die Floppy keinen Fehler meldet, erhalten Sie die Meldung '00,OK,00,00'. Falls Sie ein Programm oder mehrere Programme mit dem SCRATCH-Befehl gelöscht haben, erhalten Sie eine Information über die Anzahl der von der Diskette gelöschten Programme. Wenn die Diskettenstation durch Blinken der Lampe einen Fehler anzeigt, gibt Ihnen dieser Befehl die Art des Fehlers an. Falls Sie zum Beispiel ein Programm von der Diskette laden wollen, das nicht auf ihr existiert, erhalten Sie nach Eingabe dieses Befehls die Meldung: 62,FILE NOT FOUND,00,00. Manchmal reicht eine solche Meldung nicht aus. Dann sollten Sie im Floppy-Handbuch nachlesen. Um diese Informationen ohne DOS 5.1 zu erhalten, müssen Sie ein kleines Programm schreiben!

Der Befehl @ ersetzt folgendes Programm:

```
OPEN 15,8,15
INPUT#15,A1,A2$,A3,A4
PRINT A1,A2$,A3,A4
CLOSE 15
```

Directory

Sie können das Directory einer Diskette lesen, ohne das im Computer befindliche Programm zu zerstören. Tippen Sie einfach ein: @$ und natürlich die RETURN-Taste. Sie können auch eine bestimmte Auswahl der anzuzeigenden Programme treffen, indem Sie die Abkürzungen »?« oder »*« benutzen. Beispiel: Tippen Sie ein: @$:D* und Sie erhalten jedes Programm, das mit einem D anfängt. Die Eingabe von @$:??? listet jedes Programm, dessen Name aus genau drei Zeichen besteht.

```
**** COMMODORE 64 BASIC V2 ****

64K RAM SYSTEM  38911 BASIC BYTES FREE

READY.
LOAD"C-64 WEDGE",8

SEARCHING FOR C-64 WEDGE
LOADING
READY.
RUN

     DOS MANAGER V5.1/071382
          BY  BOB FAIRBAIRN
(C) 1982 COMMODORE BUSINESS MACHINES

READY.
```

So meldet sich das Dos 5.1

Um zum Beispiel eine Diskette neu zu formatieren, müssen Sie ohne die Benutzung des DOS 5.1 folgendes eingeben:

```
OPEN 15,8,15,"N:1541TEST/DEMO,ZX"
CLOSE 15
```

Mit dem DOS 5.1 tippen Sie ein: @N:1541TEST/DEMO,ZX

Der Befehl @N:1541TEST/DEMO ohne Angabe der ID-Nummer (in unserem Fall ZX) formatiert eine Diskette nicht neu. Er gibt der Diskette lediglich einen neuen Namen und löscht dabei das gesamte Directory. Diese Befehlsvariante läuft wesentlich schneller ab als der komplette NEW-Befehl. Allerdings muß die Diskette vorher schon formatiert gewesen sein.

Initialize

Der Befehl lautet: @ I

Er wird normalerweise nicht benötigt, kann jedoch manchmal helfen, wenn Sie ein DRIVE NOT READY erhalten.

Scratch

Wenn man Programme löschen will, tippt man ein:

@ S:Programmname

Der Befehl @ S:D* löscht alle Programmme von der Diskette, die mit D beginnen. Mit @ S:* wird die gesamte Diskette gelöscht, ähnlich dem New-Befehl, ist jedoch langsamer in der Ausführung.

RENAME

Um den Namen eines Programms auf der Diskette zu ändern, tippen Sie ein: @ R:neuer Name=alter Name

Beispiel: Aus dem Programm TEST wird das Programm LAMPE, wenn Sie eingeben: @ R:LAMPE=TEST

COPY

In der Regel wird der Copy-Befehl bei Doppellaufwerken benutzt. Er ist jedoch auch für Einzellaufwerke anwendbar. Dieser Befehl verdoppelt ein Programm oder eine Datei auf der Diskette. @ C:HUND=KATZE erstellt eine Kopie der Datei KATZE; die Kopie heißt HUND.

Eine weniger bekannte Möglichkeit des Copy-Befehls ist, daß zwei verschiedene Dateien miteinander verbunden werden können zu einer einzigen Datei. Das kann so gemacht werden:

@ C:STREIT=HUND,KATZE

Die Dateien HUND und KATZE werden zu der neuen Datei STREIT zusammengefügt. Diese Möglichkeit ist sicher sinnvoll für Dateien. Ist sie auch anwendbar auf Programme? Testen Sie es selbst.

Das DOS 5.1 belegt keinen Basic-Speicherplatz. Innerhalb des C-64 Wedge wird es geladen mit LOAD"DOS 5.1",8,1 und gestartet mit SYS 12*4096+12*256 (Schauen Sie sich das Wedge an). So können Sie es auch laden und starten. Das ist vor allem dann von Vorteil, wenn Sie ein eventuell im Speicher befindliches Programm nicht löschen wollen.

Sie sehen, daß mit diesem Programm DOS 5.1 auf der Demodiskette eine Menge anzufangen ist. Es erleichtert den Umgang mit der Floppy doch ganz erheblich. Vor allem den Directory-Befehl und den Status-Befehl wende ich selbst sehr häufig an. (gk)

Simons Basic

2 Teil

Nachdem in der letzten Ausgabe die Strukturbefehle und Programmierhilfen behandelt wurden, wollen wir nun die Grafik-, Musik-, Bildschirm- und sonstigen Befehle mit Beispielen erläutern.

Die Grafik-Befehle sollen nicht in ihrer alphabetischen Ordnung besprochen werden, sondern in ihrer natürlichen Anordnung, wie sie eventuell auch eingesetzt werden könnten. Die Grafik-Befehle sind meines Erachtens für die Programmierung der hochauflösenden Grafik eine unabdingbare Voraussetzung. Dies heißt nicht, daß diese unbedingt dem Simons Basic entnommen werden müssen, jedoch sollten diese oder ähnliche Befehle unbedingt zur Programmierung herangezogen werden.

Ihr Vorteil liegt nicht nur in der einfacheren Programmierung, sondern — und dieser Vorteil ist nicht unerheblich — auch in der schnelleren Bearbeitung. Bei komplizierten Grafiken, in denen viele Punkte gesetzt werden müssen, wartet man nicht selten mehr als eine Stunde, bis diese mit Basic erstellte Grafik fertig ist. Durch die Grafik-Befehle ergibt sich hier eine zigfach höhere Geschwindigkeit.

HIRES

Mit diesem Befehl wird im Simons Basic die Grafik eingeschaltet. Gleichzeitig werden für einfarbige Grafiken die Hintergrund- und Punktfarbe festgelegt. Vergleichen Sie hierzu Bild 1. Es zeigt ein Basic-Programm, das das gleiche bewirkt, und ein entsprechendes Assemblerprogramm, das dem Buch »Das Commodore 64-Buch, Band 1« entnommen ist. Das Basic-Programm läuft zirka zwei Minuten, wohingegen das Assemblerprogramm und auch der Befehl HIRES keine merkbare Ausführungszeit beanspruchen. Auf die verschiedenen Speicher des Commodore 64 für die Farbgrafikdarstellung soll hier nicht näher eingegangen werden, jedoch wollen wir anmerken, daß beim Einschalten der Grafik zunächst der Farbspeicher (RAM-Bereich 1023-2022) und der Grafikspeicher für die Darstellung der einzelnen Punkte (RAM-Bereich 8192-16383) mit der Farbe beziehungsweise mit Nullen vorbesetzt werden. Näheres zum Thema Grafik finden Sie in der oben angeführten Buchreihe.

MULTI

Da der Befehl HIRES nur einfarbige Grafiken zuläßt, kann man mit dem Befehl MULTI weitere drei Farben gleichzeitig auf dem Bildschirm darstellen.

LOW COL

Mit diesem Befehl sind noch weitere drei Farben für den Multi-Color-Modus zuschaltbar. Während des Programms können zwar immer nur drei Farben angesprochen werden, jedoch erscheinen bis zu sechs Farben auf dem Bildschirm.

HI COL

Mit dem Befehl HI COL können die ursprünglich mit dem Befehl MULTI dargestellten Farben im Programm wieder eingeschaltet werden. Für sechs Farben sind also die Befehle LOW COL und HI COL entsprechend im Programm zu setzen.

PLOT

Der Befehl PLOT dient zur Ausgabe eines einzigen Punktes auf dem Bildschirm. Daß dies kein einfaches Unterfangen ist, wird an dem Basic-Programm in Bild 2 deutlich. Hierzu trägt der Aufbau des Grafikspeichers erheblich bei. Ebenfalls in Bild 2 dargestellt ist die Version bei Verwendung des Befehls aus Simons Basic.

LINE

Der Befehl LINE zeichnet eine Linie mit den angegebenen Parametern auf dem Bildschirm. In Bild 3 sind wieder die entsprechenden Programme in Basic und bei Verwendung des Befehls aus Simons Basic dargestellt. Wie man an dem Basic-Programm sieht, kann man natürlich den Befehl zum Setzen eines Punktes als Unterprogramm aufrufen. Das macht aber die Ausgabe der Linie nicht wesentlich schneller. Daß es auch in einem Basic-Programm nicht einfach mit der Ausgabe einer geraden Linie getan ist, sondern auch noch einige Umwandlungen zur korrekten Pro-

```
20006 REM ********************************************
20007 REM *       G R A F I K   E I N                *
20008 REM ********************************************
20020 POKEV+17,59
20030 POKEV+24,24
20040 FORLV=1024TO2023
20050 POKELV,HF+16*ZF
20060 NEXT
20070 FORLV=8192TO16383
20080 POKELV,0
20090 NEXT
20100 POKE53280,HF
20110 RETURN

READY.
```

Bild 1a. Einschalten der hochauflösenden Grafik in Basic

```
HIRES 6,7
```

Bild 1b. Grafik einschalten mit Simons Basic

```
20400 REM ********************************************
20401 REM *       P U N K T E   S E T Z E N          *
20402 REM ********************************************
20410 IF PX<0 OR PX>319 THEN RETURN
20420 IF PY<0 OR PY>199 THEN RETURN
20430 HX=8*INT(PX/8)
20440 HY=320*INT(PY/8)+(PY AND 7)
20460 BX=2↑(7-(PX AND 7))
20470 H1=8192+HX+HY
20480 POKEH1,PEEK(H1) OR BX
20490 RETURN
PLOT PX,PY
```

Bild 2. Punkt setzen in Basic und in Simons Basic

grammausführung notwendig sind, ist aus dem Listing ersichtlich.

REC, CIRCLE, ARC, ANGL, BLOCK

Ähnlich dem Ziehen einer Linie werden durch diese Befehle ein Rechteck (REC), eine Ellipse (der Kreis ist ein Sonderfall der Ellipse-CIRCLE), Segmente (ARC), Radien (ANGL) und ausgefüllte Rechtecke (BLOCK) gezeichnet. Für Rechtecke und Blöcke können wieder die entsprechenden Befehle zum Ziehen von Linien und zum Zeichnen eines Punktes als Unterprogramme herangezogen werden, wenn das ganze in Basic programmiert werden soll. Dabei kann man einen Block als Darstellung mehrerer paralleler Linien auffassen. Für die Ausgabe von Kreisen, Ellipsen und Segmenten sowie Radien müssen die entsprechenden trigonometrischen Funktionen Sinus und Cosinus herangezogen werden. Das Zeichnen eines Radius benötigt nur zur Bestimmung des Endpunktes diese Winkelfunktionen, und ist ansonsten analog dem Ziehen einer Linie.

PAINT

Mit dem Befehl PAINT läßt sich eine vorgegebene, geschlossene Figur (Rahmen) mit einer Farbe ausfüllen. Bei entsprechend komplizierten Figuren (Konvex und konkav gebogene Randstücke, Aussparungen in der Mitte) ist das schnelle Ausfüllen eines vorgegebenen Rahmens im Basic auch ein topologisches Problem, was selbst einem einigermaßen geübten Programmierer auf Anhieb nicht gelingen wird.

DRAW, ROT

Mit dem Befehl DRAW kann eine Figur aus lauter Linien zusammengesetzt werden, und mit dem Befehl ROT kann eine derartig gezeichnete Figur um verschiedene Winkel gedreht werden.

CSET

Dieser Befehl hat gleich drei Funktionen:
- Zeichensatz umschalten (von Grafik auf Groß/Kleinschrift beziehungsweise umgekehrt)
- Grafik zurückholen
- Eine mehrfarbige Grafik zurückholen und dabei diese Grafik mit anderen Farben versehen.

CHAR, TEXT

Diese beiden Befehle sind besonders wichtig, da im Modus der hochauflösenden Grafik keine Buchstaben, Zahlen und sonstige Zeichen ausgegeben werden können. Diese beiden Befehle ermöglichen einerseits das Ausgeben einzelner Zeichen (CHAR) sowie ganzer Textzeilen (TEXT), wobei die Position, die Größe und die Farbe der Zeichen in dem Befehl selbst angewählt werden können.

Sprite-Befehle

Auch die Sprite-Befehle wollen wir in der Reihenfolge ihrer Verwendung besprechen.

DESIGN

Mit diesem Befehl wird dem Sprite Speicherplatz zugeteilt. Dies ist gegenüber dem normalen Basic keine wesentliche Erleichterung, da dies dort auch mit einem einzigen POKE-Befehl erledigt werden kann. Da man bei Simons Basic aber so gut wie gar nicht mehr auf POKE-Befehle zurückgreifen muß, ist dieser Befehl doch sinnvoll.

@

Nachdem der Speicher für ein Sprite zugeteilt wurde, muß als nächstes dieses definiert werden. Dies kann man durch 21 Zeilen, denen ein Klammeraffe vorangestellt ist und die jeweils eine Zeile des Sprites (12 oder 24 nebeneinanderliegende Punkte) enthalten.

CMOB

Dieser Befehl definiert die Farben eines Sprite.

MOB SET

Im Simons Basic bedeutet die Abkürzung MOB Moveable Objekt Block. Mit dem Befehl MOB SET werden die Eigenschaften eines Sprites festgelegt. Das sind die Farbe des Sprites, die Priorität gegenüber dem Hintergrund und ob hochauflösende Grafik oder Multicolor Grafik gewünscht wird.

MMOB, RLOCMOB

Diese beiden Befehle dienen zur Darstellung beziehungsweise zur Bewegung des Sprites. Dazu können Start- und Zielposition des Sprites angegeben werden, sowie die Geschwindigkeit mit der sich das SPRITE über den Bildschirm bewegen soll.

MOB OF

Hiermit wird das Sprite wieder ausgeschaltet.

```
20600 REM ****************************************
20601 REM *     L I N I E N   Z I E H E N        *
20602 REM ****************************************
20610 IF EX=AX OR EY=AY THEN H2=1:GOTO20640
20620 IF ABS(EY-AY) < ABS(EX-AX) THEN H2=1: GOTO 20640
20630 H2=ABS(EX-AX)/ABS(EY-AY)
20640 IFEX<AXTHENH2=H2*-1
20650 H3=1
20660 IFEY<AYTHENH3=-1
20670 IFEX=AXTHEN20740
20680 FORLL=AXTOEXSTEPH2
20690 PX=INT(LL+.5)
20700 PY=((LL-AX)*(EY-AY)/(EX-AX))+AY
20710 GOSUB20400
20720 NEXT
20730 RETURN
20740 PX=AX
20750 FORPY=AYTOEYSTEPH3
20760 GOSUB20400
20770 NEXT
20780 RETURN
LINE AX,AY,EX,EY
```

Bild 3. Linie ziehen in Basic und in Simons Basic

DETECT, CHECK

Diese beiden Befehle dienen zur Vorbereitung einer Kollisionsprüfung und zur Kollisionsprüfung selbst. Auch diese beiden Befehle sind in Basic durch einen einfachen POKE-Befehl sehr schnell zu realisieren, sie ersparen einem aber die umständliche Suche, welches Byte im Video-Controller-Chip die entsprechende Aufgabe wahrnimmt, und das Auseinanderziehen der einzelnen Bits.

Musik-Befehle

Die Anwendung der Musik-Befehle und ihrer Eigenschaften werden am besten deutlich, wenn man sich das Beispiel aus dem Handbuch ansieht (Bild 4).

Befehle für Zeichenreihen

Die Befehle für Zeichenreihen ergeben sich aus der Befehlsübersicht im ersten Teil. Wir wollen hier nur zwei Befehle besonders hervorheben:

PLACE

Der Befehl PLACE wird immer da verwendet, wo eine kleinere Zeichenreihe in einer größeren gesucht wird. Dieser wichtige Befehl fehlte bisher bei allen Commodore Basic-Generationen. Sicherlich hat jeder Programmierer, der regelmäßig Programme schreibt, sich für diesen Befehl schon ein eigenes Unterprogramm geschrieben.

USE

Mit dem Befehl USE wird den Programmierern ein noch größerer Gefallen getan als mit dem Befehl INST. In vielen Programmen wünscht man sich eine formatierte Ausgabe von Zahlen. Bisher mußte dazu immer mühsam ein Basic-Unterprogramm geschrieben werden, das eine Zahl als Zeichenreihe behandelt und in eine kaufmännische Zahlendarstellung umwandelt. Was bei anderen Basic-Dialekten selbstverständlich ist, wurde für den Commodore Computer in Simons Basic realisiert.

Befehle für Zahlen

EXOR, MOD, DIV, FRAC, % und $

Die sechs auf Zahlen anwendbaren Befehle sind wohl für jeden Programmierer im mathematisch-technisch-wissenschaftlichen Bereich unentbehrlich. Dabei ist der Befehl EXOR kein eigentlicher Zahlenbefehl. Er bildet im Prinzip nur eine weitere logische Verknüpfung neben den schon vorhandenen UND, ODER und NOT. Trick-Programmierer benutzen diesen Befehl um zwei Zahlen bzw. Bit-Muster zu vertauschen ohne Zuhilfenahme einer dritten Variablen.

Der Befehl MOD ergibt den Rest einer Zahl nach einer Division (Mod (30,4) = 2, das heißt, 30/4 = 7 REST 2) DIV entspricht dem Basic-Befehl INT, kann jedoch nur die Vorkommazahl nach einer Division bestimmen (DIV (30,4) = 7)

FRAC hingegen ergibt die Nachkommastellen einer Zahl (FRAC (3.56) = .56)

Mit % wird eine Binärzahl in eine Dezimalzahl umgewandelt und $ bewirkt das Gegenteil.

Gerade bei den manigfaltigen Adressen im Video-Controller des Commodore 64, wo einzelne Bits in Registern gesetzt oder gelöscht werden müssen, um bestimmte Tätigkeiten zu erreichen oder zu unterlassen, sind diese Funktionen sehr nützlich.

Wie zu Beginn bereits erwähnt, werden die Befehle zur Bildschirmsteuerung von Programmierern verwendet, die auch das letzte aus ihrem Computer herausholen wollen. Simons Basic bietet dazu sehr viele Möglichkeiten.

FLASH, OFF, BFLASH, BFLASH 0

Diese Befehle bewirken das Blinken von Bildschirmfarben bzw. des Rahmens. Auch die Blinkfrequenz kann variiert werden.

FCHR, FCOL, FILL, INV

Mit diesen vier Befehlen können bestimmte Bildschirmbereiche mit Zeichen und/oder Farben gefüllt bzw. invertiert werden.

MOVE

Der MOVE-Befehl dupliziert Bildschirmbereiche. Bei geschickter Bildschirmausgabe durch Cursorsteuerung läßt sich auf diese Art und Weise auch eine fensterweise Ausgabe wie bei den Computern der Serie 8000 erreichen. In diesem Falle können sogar die Fenster in einen anderen Bereich kopiert werden. Viertelt man zum Beispiel den Bildschirm, so kann man ein Viertel für die normale Bildschirmausgabe verwenden, und der Anwender kann sich bei Bedarf diese Bildschirmausgabe in ein anderes Viertel kopieren um eventuell Datenvergleiche durchzuführen.

LEFT, RIGHT, UP, DOWN

Simons Basic erlaubt weiterhin Bildschirmbereiche zu rollen. Dies kann analog zu den oben angeführten Befehlen nach rechts, links oder nach oben und unten geschehen. Sehr interessant ist zum Beispiel, daß auch Bereiche gerollt werden können, so daß der Bildschirm teilweise erhalten bleibt und in anderen Bereichen des Bildschirms fortlaufend Daten angezeigt werden können. Das Bildschirmrollen kann sowohl zyklisch als auch durch Nachziehen von Leerzeilen bzw. -spalten erfolgen.

```
100 REM ********************************
110 REM *** BEISPIEL!                ***
120 REM *** MUSIK                    ***
130 REM ********************************
140 REM *** LAUTSTAERKE EINSTELLEN   ***
150 VOL 10
160 REM *** WELLENFORM EINSTELLEN    ***
170 WAVE 1,00010000
180 REM *** HUELLKURVE EINSTELLEN    ***
190 ENVELOPE 1,8,8,8,0
200 REM *** FESTLEGEN DER NOTEN      ***
210 A$= "■IZ■C5■E5■F5■"
220 A2$="G5■C5■E5■F5■G5■C5■E5■F5■G5■"
230 A2$=A2$+"E5■C5■E5■D5■E5■E5■D5■C5■"
240 A2$=A2$+"C5■E5■G5■G5■F5■F5■E5■F5■"
250 A2$=A2$+"G5■E5■C5■D5■C5■C5■C5■E5■"
260 A2$=A2$+"F5■"
270 A3$="C5■C5■"
280 MUSIC 8,A$+A2$+A2$+A3$
290 REM *** MUSIK SPIELEN            ***
300 PLAY 1
310 REM *** LAUTSTAERKE 0            ***
320 VOL 0
330 END

Ergebnis:

        "WHEN THE SAINTS GO MARCHING IN"

wird gespielt.
```

Musik mit Simons Basic (entnommen aus dem Handbuch)

SCRSV, SCRLD

Bildschirminhalte im normalen Modus können mit diesen beiden Befehlen auf Diskette gespeichert beziehungsweise wieder geladen werden. Dies kann zum Beispiel interessant sein, wenn auf dem Bildschirm Balkengrafiken dargestellt wurden, deren Daten erst mühsam errechnet werden mußten. Zur nochmaligen Anzeige eines solchen Diagramms braucht dann keine neue Berechnung durchgeführt, sondern lediglich der Bildschirminhalt von Diskette geladen werden.

COPY, HRDCPY

Mit den beiden Copy-Befehlen sind sowohl die Bildschirmausgabe von hochauflösender Grafik (COPY) als auch eines Bildschirmes im Normalmodus (HRDCPY) auf einen Drucker möglich. Dies sind zwei interessante Befehle, besonders das Hardcopy der hochauflösenden Grafik, da dies eine relativ umständliche Druckerprogrammierung erfordern würde.

Befehle für LIGHTPEN, JOYSTICK und PADDLE

Mit den vier Befehlen zur Abfrage des Status der obengenannten externen Hilfsgeräten lassen sich sicherlich sehr einfach Spiele programmieren. Statt umständlicher IF-Abfragen im Programm können die Werte der externen Geräte durch diese Befehle sofort erfragt und somit auch gleich weiter verarbeitet werden.

Sonstige Befehle

DESIGN

Ähnlich den Sprites können auch die Zeichen des normalen Zeichensatzes neu definiert werden. Mit dem Befehl DESIGN wird zunächst festgelegt welches Zeichen erstellt werden soll und mit dem Befehl @ wird dieses Zeichen auch ähnlich wie bei den Sprites kreiert.

DIR, DISC

Mit dem DISC-Befehl können Befehle an die Floppystation unmittelbar übergeben werden, ohne OPEN, CLOSE und die entsprechenden Fehlerkanäle anzugeben. Mit DIR kann das Inhaltsverzeichnis einer Diskette ausgegeben werden, wobei auch Jokerzeichen zugelassen sind, so daß ein sogenanntes »Pattern Matching« möglich ist. Dies bedeutet, daß man sich Programme oder Dateien, die ganz bestimmte Buchstabenkombinationen enthalten, auszugsweise auflisten lassen kann.

FETCH

Der FETCH-Befehl ermöglicht es, kontrollierte Eingaben im Programm zuzulassen, ohne daß aufwendige INPUT-Routinen geschrieben werden müssen.

INKEY

Sehr wichtig ist auch der INKEY-Befehl, da er abfragt ob eine Funktionstaste gedrückt ist. Dies ermöglicht eine schnelle Bearbeitung durch den Anwender, da nicht immer RETURN nach einer Eingabe gedrückt werden muß, beziehungsweise eine umständliche Überprüfung über den GET-Befehl entfällt.

LIN

Normalerweise beginnt eine Routine zum Positionieren des Cursors immer in der linken oberen Bildschirmecke und steuert die gewünschte Cursorposition dann durch eine entsprechende Anzahl von »Cursor Right« und/oder »Cursor Down« an. Dies muß bei absolutem Cursorpositionieren auch dann erfolgen, wenn die Ausgabe bereits in der nächsten Zeile erfolgen soll. Mit dem LIN-Befehl ist auch eine relative Zeilenpositionierung des Cursors möglich.

MEM

Für Anwender, die ihren Zeichensatz selbst programmieren beziehungsweise diverse Zeichen verändern wollen, ist der MEM-Befehl interessant, da er ohne komplizierte USR-Funktion den ROM-Bereich des Zeichensatzes in den entsprechenden RAM-Bereich verlegt.

PAUSE

Sicherlich hat jeder Programmierer in seinem Programm irgendwo eine »leere« FOR...NEXT-Schleife, um eine Anzeige eine gewisse Zeit am Bildschirm aufrecht zu erhalten.

Einerseits ist das ein Programmiertrick, der die Dokumentation des Programms nicht wesentlich erleichtert, andererseits ist in diesen Schleifen nur eine ungenaue Zeitangabe möglich. Mittels des PAUSE-Befehls kann man nun das Programm für eine genau definierte Anzahl von Sekunden anhalten und sogar zusätzlich noch eine Meldung drucken.

RESET

Der RESET-Befehl ist nicht zu verwechseln mit irgendwelchen RESET-Tasten und/oder -Schaltern. In Simons Basic dient er zum Setzen eines Zeigers auf eine beliebige DATA-Zeile. Dies wird dann benötigt, wenn in einem Programm mehrere verschiedene DATA-Blöcke vorkommen, die teilweise oder ganz neu eingelesen werden müssen.

Zusammenfassung

Als Ganzes gesehen ist Simons Basic mit Sicherheit eine sinnvolle Basic-Erweiterung für den Commodore 64. Erfahrene Programmierer können die Befehle von Simons Basic sogar in den meisten Fällen ohne genaues Studium des Handbuches verwenden, da die Syntax der Befehle vollkommen ausreichend zu ihrer Erklärung ist. Jedoch empfiehlt es sich zu Beginn ein bißchen mit den Befehlen herumzuspielen. Auch für den Anfänger ist Simons Basic eine wertvolle Hilfe, jedoch dürfte für diesen es Anfangs neu sein, daß auch Basic-Befehle Parameter erhalten können, was zum Beispiel in hohem Maße bei den Grafik- und Sprite-Befehlen der Fall ist.

Die unter Programmierhilfen zusammengefaßten Befehle bilden eine weitgehend ausreichende Basis um Programme sehr schnell und methodisch austesten zu können beziehungsweise die Stuktur-Befehle ermöglichen eine übersichtlichere Programmstruktur, die sich dann einfacher dokumentieren läßt.

Insbesonders erwähnenswert scheinen mir auch die Programmschutz-Befehle, die zwar an sich keinen eigenen Programmschutz darstellen (unerlaubtes Kopieren ist immer noch möglich), aber in Verbindung mit anderen Schutzmaßnahmen sicherlich einen ausreichenden Schutz gegen unbefugte Programmnutzung darstellen können. Man denke hier nur an ein Passwort, was im Programm unkenntlich gemacht wird, so daß es nicht einfach dem Programm-Listing zu entnehmen ist.

Das Handbuch ist zwar klar und übersichtlich gegliedert und mit ausreichend erklärenden Beispielen versehen, jedoch wären manche Übersichtstabellen (wie zum Beispiel eine Zusammenfassung aller Befehle nach Bereichen geordnet mit Angabe ihrer Syntax/Übersicht von Befehlen mit Parametern) als Nachschlagewerk für den fortgeschrittenen Programmierer wünschenswert.

Fazit: Simons Basic ermöglicht es, den Commodore 64 effizient in allen seinen Möglichkeiten zu nutzen.

(H.-L. Schneider)

Superbase 64, mehr als eine »SUPER«-Datenbank? Vom Hersteller wird das Produkt als komplettes Informations-Kontrollsystem für den Commodore 64 bezeichnet. Es wird zum Preis von zirka 300 Mark mit einer Systemdiskette, einer

Superb

Bild 2. Das Hauptmenü

Bild 1. Die Maske

Bild 4. Die Kundendatei

Das Handbuch ist sehr übersichtlich aufgebaut und führt im ersten Teil, dem sogenannten Tutorium, in kleinen Schritten von der Erläuterung der Tastatur bis zur komplexen Fakturierung. Die einzelnen Schritte sind verständlich beschrieben, leider nicht in deutsch!

Im zweiten Teil des Handbuches werden alle Kommandos ausführlich erläutert, auch die Funktionen, die im Tutorium nicht angesprochen werden. Grundsätzlich kann mit Superbase 64 ohne irgendwelche Programmierkenntnisse gearbeitet werden. Wer jedoch tiefer einsteigen will und sich mit den »normalen« Funktionen noch nicht zufrieden gibt, bekommt im dritten Teil des Handbuches genügend Tips und Tricks zum Experimentieren.

Von den Anwendungsmöglichkeiten zerfällt das Softwarepaket in zwei Bereiche:

1. Datenbank-Management-System
2. Kalkulation und Fakturierung

Der Aufbau und die Verwendung einer Superbase-Datenbank wird nachfolgend anhand eines Schallplattenarchivs im einzelnen aufgezeigt. Im Anschluß daran zeigt ein anderes Beispiel die Möglichkeit zur Kalkulation und Rechnungserstellung auf.

Schallplatten-Archiv mit Superbase 64

Um mit Superbase 64 arbeiten zu können, braucht man einen Commodore 64 und ein Diskettenlaufwerk VC 1541. Im Gegensatz zu manch anderem Dateiverwaltungsprogramm genügt hierzu wirklich ein einziges Diskettenlaufwerk, da das Programm komplett geladen wird, beziehungsweise die erforderlichen externen Teile mit auf die Datendiskette kopiert werden. Das erfordert zwar etwas mehr Vorarbeit, macht aber den späteren Arbeitsablauf angenehm und erfor-

Bild 3. Das Plattenarchiv

```
Mode  Select n,l,p,f,k,c,m,o,a,r,d,+-
        Schallplattenarchiv
Kennbegriff:  <Bach, Johann Seb.
Plattentitel: <Violinkonz. Bwv 1041-43>
Komponist:    <Bach, Johann Seb.        >
Orchester:    <Moskauer Kammerorchest.>
Dirigent:     <Rudolf Barschai          >
Besetzung:    <Oistrach, Violine        >
Stilrichtung: <Klassik                  >
Plattenummer: <25931 HK   > TYP:<LP     >
Hersteller:   <Ariola     > JAHR: <1963>
Notizen 1:    <                         >
Notizen 2:    <                         >
```

...base 64

Backup-Diskette und einem englischen Handbuch ausgeliefert.

Bild 5. Die Rechnungsdatei

dert keine Diskjockey-Fähigkeiten. Da grundsätzlich alle Datenbankabfragen und -auswertungen sowohl über Drucker als auch über Bildschirm möglich sind, wird der Drucker nicht unbedingt benötigt, um den vollen Funktionsumfang nutzen zu können.

Mit LOAD"SB",8,1 wird in zirka 2 Minuten zunächst das Programm geladen. Danach wird die Systemdiskette gegen eine Datendiskette ausgetauscht. Auf diese werden zunächst die Hilfsmasken (help screens) und das Demonstrationsprogramm »Training« kopiert; Dauer zirka 6 Minuten. Danach wird erneut das Programm von der Systemdiskette geladen und anschließend die Datendiskette eingelegt.

Diese Vorarbeit von zirka 10 Minuten stellt keinen besonderen Aufwand dar, bietet aber den unschätzbaren Vorteil, daß für alle folgenden Arbeiten keinerlei Diskettenwechsel mehr notwendig sind.

Grundsätzlich kann eine Superbase-Datenbank bis zu 15 Einzeldateien umfassen. Ein Datensatz kann maximal 1108 Zeichen lang sein und auf 4 Bildschirmseiten verteilt werden. Pro Satz sind maximal 127 Felder inklusive Schlüsselfeld möglich, wobei das Schlüsselfeld maximal 30

Fortsetzung auf Seite 50

superbase 64

Fortsetzung von Seite 47

Stellen lang sein kann; normale Textfelder können bis zu 255 Zeichen beinhalten und sich somit auch über mehrere Zeilen erstrecken.

Wir wollen nun eine Schallplatten-Archivdatei einrichten und dabei die wichtigsten Funktionen aufzeigen. Wir geben auf die Frage nach dem Programmnamen »Training« ein und beantworten die Aufforderung »Enter Filename« mit der Eingabe »Archiv«. Danach erscheint links oben die Meldung »Mode: Format«, das heißt wir befinden uns im Formatierungsmodus. Auch bei allen weiteren Operationen wird links oben der jeweilige Modus angezeigt, was eine angenehme Orientierungshilfe darstellt.

Bildschirmmaske definieren

Auf dem im übrigen leeren Bildschirm werden nun die einzelnen Maskenfelder definiert. Dies läuft im einzelnen folgendermaßen ab:

— Feldnamen eingeben
— Cursor auf Feldanfang positionieren
— f1-Taste drücken und Buchstaben für Felddefinitionen (zum Beispiel »K« = Key, »T« = Text, »D« = Datum) eingeben
— Feld mit Cursor durchlaufen, bis der Stellenzähler rechts oben die gewünschte Feldlänge anzeigt
— Feldende mit RETURN-Taste markieren

Alle Felder sind nun am Feldanfang mit einem kleinen ausgefüllten Quadrat und am Feldende mit einem größeren schraffierten Quadrat gekennzeichnet. Mit f1-Taste und »I« können Felder invertiert dargestellt werden, mit f1-Taste und »S« wird der gesamte Bildschirm invertiert. Nachträgliche Änderungen sind etwas umständlicher durchzuführen: Mit f1-Taste und »E« (= Error) wird nur der Feldname gelöscht; um das Feld selber zu löschen, muß der Cursor genau auf den Feldanfang positioniert werden und nochmals f1-Taste und »E« eingegeben werden.

Wenn die Maske endgültig definiert ist, wird sie mit der f1-Taste und der RUN/STOP-Taste abgespeichert. Gleichzeitig werden die bisherigen Feldbegrenzungszeichen in » < « und » > « umgesetzt, so daß die Maske nun so aussieht wie in Bild 1 dargestellt.

Menü-Steuerung

Die Frage »DUPLIKATE KEYS ALLOWED?« wird mit »y« oder »n« beantwortet, je nachdem, ob man gleiche Hauptschlüssel zulassen will oder nicht. Danach meldet sich das Programm mit dem Hauptmenü 1. Von jetzt an wird das Arbeiten mit Superbase 64 immer mehr zum Vergnügen. Mit RETURN kann man einfach in ein Hauptmenü 2 (Bild 2) umsteigen. Die jeweils gewünschte Funktion kann wahlweise über die entsprechende Funktionstaste oder die Eingabe in der Kommandozeile ausgelöst werden. Mit der f8-Taste kann aus dem Hauptmenü 1 jederzeit eine HELP-Funktion aufgerufen werden. Die Beschreibung der gewünschten Funktion kann hiermit abgerufen werden. Außerdem können auch eigene HELP-Masken definiert werden.

Daten eingeben

Über Hauptmenü 1 und der f1-Taste wird die ENTER-Funktion gestartet. Sofort erscheint eine leere Bildschirmmaske und links oben »Mode Entry«. Das Schlüsselfeld ist ein absolutes Mußfeld. Die anderen Felder sind wahlweise, sofern sie nicht mit F (= forced) definiert sind. Mit RETURN kommt man an den Anfang des nächsten Feldes, mit HOME an den Maskenanfang. Drückt man nach dem Ausfüllen des letzten Feldes nochmals RETURN, so wird der Satz sofort abgespeichert. Werden nicht alle Felder ausgefüllt, so kann das Abspeichern schon vorher mit SHIFT/RETURN ausgelöst werden. Mit der SPACE-Taste bekommt man das nächste Leerbild. Das Duplizieren von Sätzen ist über »SELECT« und »A« (= add) möglich. Die Dateneingabe ist schnell und denkbar einfach.

Datenbank-Abfragen

Hier bietet Superbase 64 nahezu unbegrenzte Möglichkeiten. Der schnellste Zugriff erfolgt über das Schlüsselfeld (Key). Die Möglichkeiten der »SELECT«-Funktion reichen vom beliebigen Blättern in der Datenbank bis zu kompliziertesten UND- beziehungsweise ODER-Verknüpfungen, verbunden mit Vergleichsoperationen wie größer, kleiner oder ungleich (>, <, #). Suchen mit Joker ist genauso problemlos möglich wie Suchen über Matchcode.

Die aus der gesamten Datei ausgewählten Sätze können nacheinander am Bildschirm angezeigt werden. Sehr angenehm ist es auch, daß man jederzeit auf nicht ausgewählte »Nachbar-Sätze« springen kann. Die ausgefilterten Ergebnisse zu einer Abfrage können auch in einer Hilfsdatei abgelegt werden. Die Selektionsmöglichkeiten von Superbase 64 lassen absolut keine Wünsche unbefriedigt.

Sortierte Datenbankauswertungen

Durch optimales Zusammenwirken der Funktionen »SELECT«,

»SORT« und »OUTPUT« lassen sich beliebige Datenbankauswertungen in übersichtlicher Form darstellen. Die Ausgabe kann wahlweise über Bildschirm oder Drucker erfolgen. Wer nicht unbedingt ein schriftliches Protokoll benötigt, kann auch ohne teuren Drucker fast alle Vorteile von Superbase 64 voll nutzen.

Am Beispiel des Schallplattenarchivs könnte eine sehr einfache Datenbankauswertung folgendermaßen aussehen:

— Mit »SELECT« und »M« (Matchcode) werden zunächst bestimmte Plattentitel ausgewählt und in der Datei »HLIST« abgelegt.

— Mit »SORT« und »FROM 'HLIST' ON (Plattentitel)« werden diese Ergebnisse nach Plattentitel umsortiert.

— Mit »OUTPUT« und »FROM 'HLIST' (Plattentitel)« werden alle Plattentitel (ohne andere Felder) am Bildschirm ausgegeben (siehe Bild 3).

Daß Superbase 64 auch über einen leistungsfähigen Listenprogrammgenerator (REPORT-Funktion) verfügt, gehört nach den bisher aufgezeigten Funktionen einfach zu den Selbstverständlichkeiten.

Resümee bei der Arbeit mit Superbase 64 als Datenbanksystem: Hat man die nicht immer ganz einfachen Vorleistungen zur Erstellung der Datenbank inklusive Maskendefinition hinter sich gebracht, so macht die Dateneingabe und -pflege Vergnügen. Die Abfrage- und Auswertungsmöglichkeiten sind nahezu unbegrenzt, das heißt, man kann zu Recht von einer »SUPER«-Datenbank sprechen.

Kalkulation und Fakturierung

Die vielfältigen Funktionen von Superbase 64 als Datenbanksystem waren schon sehr beeindruckend. Darüber hinaus enthält dieses Programm jedoch noch Kalkulationsmöglichkeiten, wie sie sonst oft nur von einem separaten Programm abgedeckt werden. Damit werden auch Anwendungsgebiete wie Fakturierung und Verkaufsstatistik erschlossen. Auch diese Möglichkeiten sollen wieder anhand eines konkreten Beispiels aufgezeigt werden.

Kunden-Datei aufbauen

Zunächst wird eine Kundendatei aufgebaut. Neben der Kunden-Nr. 9.4. (= Schlüssel), Kundenname, Telefon und Anschrift werden auch Felder wie Kredit-Limit, Datum der letzten Rechnung, Datum der letzten Zahlung und Saldo aufgenommen (siehe Bild 4). Der Aufbau dieser Datei erfolgt im Prinzip genauso wie beim Beispiel des Schallplattenarchivs, so daß hier auf nähere Erläuterungen verzichtet werden kann.

Rechnungsdatei aufbauen

Als nächstes wird eine Rechnungsdatei aufgebaut mit Rechnungsnummer (= Schlüssel), Kundennummer, Datum der Rechnungsserstellung, Lieferung (Stückzahl und Artikel), Preis, Betrag, Rabatt, Summe, Mehrwertsteuer und Endsumme (siehe Bild 5). Diese Maske beinhaltet sowohl Felder, die jeweils ausgefüllt werden müssen, als auch solche, deren Inhalt errechnet wird (Betrag, Summe, Endsumme). Diese Felder sind bei der Maskenerstellung mit »R« (= result) definiert worden und beinhalten die jeweilige Rechenformel. Diese lautet zum Beispiel für das Feld Summe: (Betrag)-(Betrag)*(Rabatt)/100.

Sehr angenehm in der Bedienung ist es, daß bei numerischen Eingabefeldern automatisch die Buchstaben auf der Tastatur gesperrt sind, was Eingabefehler verhindert. Etwas ungewohnt ist die Form von Datumsfeldern, die alle in der Form »JAN1584« ausgefüllt werden. Auf eine Besonderheit soll noch hingewiesen werden: Das Feld Lieferung ist als ein durchgehendes Textfeld definiert. Am Feldanfang wird ein numerischer Wert, nämlich die Stückzahl eingegeben, danach durch Leerstelle getrennt die Artikelbezeichnung. Bei der Kalkulation erkennt das Programm selbst, daß es mit dem Wert »2« rechnen soll.

Verkaufsstatistik

Bedient man sich nun wieder der Funktion »OUTPUT«, so kann man sich mit der Angabe ALL (Lieferung) (Betrag) eine komprimierte Übersicht über alle verkauften Artikel, Stückzahlen und Beträge ausgeben lassen. Natürlich könnte diese Übersicht zum Beispiel auch nach Artikelbezeichnung sortiert werden. Eine Verkaufsstatistik auf Bildschirm oder Liste ist also kein Problem.

Dienstprogramme

Der gute Eindruck von einem soliden und leistungsfähigen Software-Produkt wird noch abgerundet durch einige nützliche Dienstprogramme. Unter anderem kann mit Catalog jederzeit der Superbase-interne Katalog angegeben werden beziehungsweise mit Directory der Inhalt der gesamten Diskette. Mit Backup können schnell Sicherungskopien der Datendiskette erstellt werden. Besonders muß noch auf die Funktionen Import/Export hingewiesen werden:

— Mit Import können Daten von externen Programmen eingelesen und mit Superbase-Dateien zusammen weiterverarbeitet werden.

— Mit Export können Daten aus Superbase-Dateien als sequentielle Datei ausgegeben werden, die dann von anderen, zum Beispiel Basic-Programmen, weiterverarbeitet werden können.

Abschließende Beurteilung

Für den günstigen Preis von zirka 300 Mark bekommt der Käufer ein sehr bedienungsfreundliches Software-Paket, das so viele Möglichkeiten bietet, wie man sie sonst oft nur mit mehreren Einzelprogrammen geboten bekommt. Die Funktionen der »SUPER«-Datenbank sind beeindruckend. Die zusätzlichen Kalkulationsmöglichkeiten eröffnen ein breites Anwendungsgebiet, das weit über reine Datenverwaltungen hinausgeht. Besonders angenehm sind auch die komprimierten Auswertungen wahlweise auf Bildschirm oder Drucker. So bleibt mit Superbase 64 kaum ein Wunsch unerfüllt, höchstens der nach einem Handbuch in deutscher Sprache.

(Arnd Wängler)

Um jeweils 100 Mark unterscheiden sich die drei Datenverwaltungsprogramme Datamat, Datenmanager und Multidata. Was diese einzelnen Programme leisten und wodurch der Preisunterschied gerechtfertigt ist, haben wir getestet.

Daten gut im Griff

Datamat, Multidata und Datenmanager

Textverarbeitung mit Multidata

Hauptmenü von DATAMAT

Um es vorwegzunehmen: Wer längere Listen in passablen Zeiten zu Papier bringen will (oder muß), ist mit Datamat nicht allzu gut beraten. In diesem Punkt ist Datamat von Data Becker den heutigen Anforderungen in punkto Büroeinsatz keineswegs gewachsen. Wer aber vor allem Wert auf viele Sortiermöglichkeiten legt, erhält für 99 Mark ein recht vielseitiges, flexibles Dateiverwaltungsprogramm.

Die Anleitung besteht aus zwei Teilen: Einem ausführlichen Einführungs- und Erläuterungsteil und einer knappen Zusammenfassung. Schon aus der Aufmachung der Anleitung geht eigentlich hervor, daß sich dieses Programm vor allem am Einsteiger orientiert, so daß erfahrene Anwender erst recht gut mit Datamat zurechtkommen sollten.

Was die Arbeitsgeschwindigkeit anbelangt, so werden jene, die schon mit teuren »professionellen« Dateiverwaltungsprogrammen gearbeitet haben, enttäuscht sein. Datamat ist zum großen Teil in Basic geschrieben, was die Programmbearbeitung nicht gerade beschleunigt. Das Preis/Leistungsverhältnis insgesamt gesehen muß man trotzdem als recht gut bezeichnen.

Programmbeschreibung

Für Datamat zur Verfügung stehen muß auf jeden Fall das Floppy-Laufwerk VC 1541 und, wenn man drucken will, ein Drucker. Will man andere Drucker als den VC 1525 oder den VC 1526 verwenden, wie zum Beispiel Epson-Drucker, so können diese im Menüteil »Dienstprogramme« mit Hilfe einer »Druckertabelle« angepaßt werden.

Nach dem Laden meldet sich »Datamat« mit dem Hauptmenü. Die einzelnen Programmteile lassen sich mit Hilfe der Cursortasten anwählen. Die Funktionstasten sind mit Befehlen wie »Zeilen löschen«, »einfügen« und »abspeichern« be-

legt. Man benötigt immer zwei Disketten (Daten- und Programmdiskette), was die Bedienung recht umständlich macht, falls man nicht zwei Floppy-Laufwerke zur Verfügung hat.

Die Eingabe von Daten gestaltet sich recht einfach mit Hilfe der Cursor- und der Returntaste.

Das Suchen bestimmter Daten kann auf recht vielseitige Weise erfolgen:

Hauptmenü von MULTIDATA

Untermenü von MULTIDATA

a) nach Indexfeld;
b) über Index mit *;
c) über andere Felder als das Indexfeld, wobei mehrere Felder als Kriterium gewählt werden können. (Dies ist aber sehr zeitaufwendig!)
d) Kombinierte Suche mit Indexfeld und anderen Feldern.

Das Suchen ist also recht einfach und komfortabel, nimmt aber selbstverständlich um so mehr Zeit in Anspruch, je mehr Kriterien man angibt.

Ebenso einfach ist das Ändern und Löschen von Datensätzen.

Das Menü

1. Datei einrichten

Dieser Programmteil dient dem Erstellen der Eingabemaske. Es können bis zu 50 Eingabefelder definiert werden. Ein einzelnes Feld kann dabei die Länge von minimal zwei Zeichen und maximal 40 Zeichen haben. Eine Zeile kann mehrere Felder enthalten. Die Länge eines kompletten Datensatzes darf bis zu 253 Zeichen umfassen. Die Eingabemaske kann unabhängig der Dateizugehörigkeit abgespeichert werden und kann, falls sinnvoll, für verschiedene Dateien genutzt werden. Das Programm errechnet nun aus der Größe des Datensatzes und des Indexfeldes die maximale Anzahl der Datensätze. Die folgende Speicherplatzreservierung nimmt oft mehrere Minuten in Anspruch.

2. Daten pflegen

Dieser Programmteil dient der Erfassung, der Änderung und dem Löschen von Daten. Datamat bietet die Möglichkeit, jederzeit eine Hardcopy vom Bildschirm zu erstellen, womit der Ausdruck von gesuchten Datensätzen gegeben ist. Doch leider ist gerade diese Möglichkeit gleichzeitig ein Schwachpunkt. Denn hat man versehentlich den Drucker nicht angeschlossen, stürzt das Programm kurzerhand ab und man darf neu beginnen.

3. Datei sortieren

Die zu suchenden Datensätze können nach mehreren Kriterien ausgewählt und sortiert werden. Sucht man in einer Adreßdatei zum Beispiel nach »R*« so werden alle Namen, die mit »R« beginnen ausgegeben. Ebenso kann das »R« als Ober- oder Untergrenze eingegeben werden, wobei dann von »A« bis »R« oder von »R« bis »Z« gesucht wird. Es lassen sich aber auch Namen oder Buchstaben ausschließen, die nicht berücksichtigt werden sollen.

4. Datei auswerten

Auch dieser Programmteil bietet recht gute Möglichkeiten. Das Aussehen einer Liste kann frei gewählt werden, in dem man die einzelnen Felder auf andere (freie) Stellen verschiebt. Das Drucken von Adreßetiketten ist natürlich auch hier möglich, was eigentlich immer bei »professionellen Datenverarbeitungsprogrammen« vorausgesetzt werden sollte. Darüber hinaus wird die Möglichkeit geboten, Steuerzeichen zu senden, die zum Beispiel den Schrifttyp festlegen.

Hier gibt es also vielseitige Möglichkeiten, auf die aber in der Anleitung nicht deutlich genug hingewiesen wird. So gibt es auch eine Schnittstelle zum Textverarbeitungsprogramm Textomat, doch wird auf diese auch nur sehr vage hingewiesen.

5. Programme beenden

Leider ein weiterer Schwachpunkt. Im Teil eins der Anleitung wird nicht deutlich darauf hingewiesen, daß dieser Programmteil immer durchlaufen werden muß,

Daten gut im Griff

Datamat,

bevor man den Commodore 64 ausschaltet. Sonst geht die Indexdatei verloren, was schlicht zur Folge hat, daß die ganze Datei als nicht vorhanden gilt. Es besteht zwar die Möglichkeit den Index zu erneuern, und seine Datei zu retten (siehe Dienstprogramme), aber ob dies die optimale Lösung ist, sei dahingestellt.

6. Dienstprogramme

Dieser Programmteil bietet folgendes:

a) Man hat die Möglichkeit, einen vorhandenen Index zu erneuern oder einen bestehenden Index zu ändern, falls zum Beispiel ein anderes Feld als das ursprüngliche zum Indexfeld erklärt werden soll.

b) Durch Erstellen einer »Druckertabelle«, die eine Neudefinition des Zeichensatzes, eine eventuellen Änderung der Sekundäradresse und noch einiges mehr beinhaltet, soll es möglich sein, fast alle beliebigen Druckertypen zu verwenden.

Multidata

Multidata von Commodore ist vom Preis her (298 Mark) nicht mit dem Datamat zu vergleichen. Doch es geht uns um das Preis/Leistungsverhältnis, bei dem Multidata wohl besser abschneidet. Multidata hat allerdings auch ein integriertes Textverarbeitungsprogramm, um das der Datamat erst erweitert werden müßte. Wenn man also das reine Datenverwaltungsprogramm beurteilt, liegen beide Programme im Preis nicht so weit auseinander.

Das Handbuch von Multidata liefert alle Informationen, die der Benutzer benötigt. Die Führung durch das Programm wird durch das leicht zu handhabende Menü erreicht. Bevor das eigentliche Arbeiten mit Multidata beginnt, wird die Sprache abgefragt, in der man arbeiten will. So stehen Englisch, Deutsch, Französisch, Italienisch und Spanisch zur Auswahl. Nun meldet sich das Hauptmenü. Es stehen die Menüpunkte

1. Disketten Formatierung
2. Disketten Prüfung
3. Arbeit mit Dateien
4. Sortieren von Dateien
5. Listen/Aufstellung
6. Daten Übertrag
7. Textverarbeitung

zur Verfügung

Die Funktionstastenbelegung entnehmen Sie bitte dem Bild. Man kann mit ihnen zum Beispiel steuern, ob man mit akustischen Signalen arbeiten will. Sinnvoll ist dies vor allem bei längerdauernden Operationen wie dem Sortiervorgang, da sich das System dann akustisch zurückmeldet, und man nicht ständig auf den Bildschirm sehen muß.

```
Datei    Datensatz  Geloescht  Leer
Adressen    36          0       576

1 Organisation der Datei
2 Neue Datensaetze erstellen
3 Bestehende Datensaetze suchen
4 Auf andere Diskette kopieren
5 Reorganisation
6 Daten loeschen
7 Ganze Datei loeschen

Option
```

Untermenü von Multidata

Das Menü

1. Disketten Formatierung

Multidata benötigt zum Abspeichern von Daten oder Texten eine eigene Diskette, die durch diesen Menüpunkt erstellt wird. Auf dieser Diskette können dann 18 verschiedene Dateien und 18 verschiedene Texte gespeichert werden. In der Verarbeitung können Texte und Daten kompiniert werden.

Pro Datei sind maximal
- 9999 Datensätze ohne Index
- 3699 Datensätze mit einem oder mehreren Indexen
- 16 verschiedene Informationszeilen (Felder)
- 27 Zeichen in den alphanumerischen Zeilen
- 9 Ziffern (Ganze- und Dezimalstellen) für numerische Zeilen
- 9 Zeichen für Dateneinnamen
- 9 Zeichen für Zeilenbenennung

zulässig. Diese Dateien können dann im Menüpunkt drei erstellt werden.

2. Disketten Prüfung

Mit dieser Option kann man zum einen überprüfen, ob die benutzten Disketten betriebsfähig sind, zum anderen das Dateiverzeichnis der zu prüfenden Diskette lesen, um sie richtig etikettieren zu können.

3. Arbeiten mit Dateien

Wählt man diesen Programmteil, erscheint zunächst eine Auswahl zwischen

a) Erstellung von neuen Dateien
b) Arbeit mit bestehenden Dateien.

Bei der Erstellung von neuen Dateien kann man, in den oben erwähnten Grenzen, die Form einer Datei wählen, die man benötigt, wobei mehrere Indexfelder bestimmt werden können.

Wählt man die Arbeit mit bestehenden Dateien, so erscheint folgendes Untermenü:

1. Organisation der Datei
2. Neue Datensätze erstellen
3. Bestehende Datensätze suchen
4. Auf andere Disketten kopieren
5. Reorganisation
6. Daten löschen
7. Ganze Datei löschen.

Bei Organisation der Datei wird dem Benutzer gezeigt, wie der Datensatz aufgebaut ist. Die restlichen Menüpunkte sollen an dieser Stelle nicht weiter erklärt werden.

4. Sortieren von Dateien

Mit dem Sortierprogramm können die Datensätze nach bis zu neun Kriterien sortiert werden. Man kann sowohl aufsteigend als auch fallend sortieren. Diese Option ist nur für das Sortieren allgemein vorgesehen. Für die Ausgabe bestimmter Datensätze ist der folgende Menüpunkt vorgesehen.

5. Listen / Aufstellungen

Diese Option wird verwendet, um mit dem Computer jede Art von Detailinformationen auf dem Bildschirm zu erzeugen oder auszudrucken. Hier kann man wählen, welche Bereiche der Datei man aus-

Multidata und Datenmanager

werten will. Weiterhin bietet diese Option die Möglichkeit, mit einzelnen oder mehreren Feldern zu rechnen. So kann man zum Beispiel DM-Angaben in Dollar umrechnen.

6. Datenübertragung

Dieser Menüpunkt dient dem automatischen Übertragen von Informationen von einer Datei auf eine andere. So können hier ganze Seiten oder auch nur einzelne Felder übertragen werden.

7. Textverarbeitung/Etikettieren

In diesem Programmteil können unter anderem mit Multidata erstellte Texte mit Dateien kombiniert und verarbeitet werden. Wo in den Texten Daten oder Datensätze verwendet werden, bleibt dem Benutzer überlassen.

Die Text-Erstellung gestaltet sich größtenteils recht komfortabel, hat aber auch ihre Schwachpunkte. So kann man zwar die linke Seitenbegrenzung in der Zeile einstellen, doch für die rechte Begrenzung kann kein Wert eingegeben werden. Wenn man zum Beispiel die linke Begrenzung auf 10 einstellt, muß man selber berechnen, wieviel Platz noch in der Zeile bleibt. Alles in allem ist auch die Textverarbeitung, nach einiger Gewöhnung, sehr gut zu handhaben, so daß sie sich wohl auch kommerziell anwenden läßt. Beispielsweise ist auch das Erstellen von Formbriefen und deren automatisches Eintragen von Adressen vorgesehen.

Wie Commodore uns mitteilte, sind die neuesten Versionen von Multidata um einen Programmpunkt erweitert worden. Dieser Programmpunkt läßt das Kopieren von Datendisketten auch bei einem Floppy-Laufwerk zu.

Datenmanager

Der Datenmanager, (198 Mark) das jüngste Kind von Commodore, ist keine verkleinerte Form des Multidata, sondern ein eigenständiges Produkt, das aber alle Vorzüge des »großen Bruders« bietet. Auch in diesem Programm sind sowohl Datenverwaltung als auch Textverarbeitung enthalten. Das Programm verwaltet die Dateien im Direktzugriff.

Die wichtigsten Daten:

Maximale Anzahl von Datensätzen pro Diskette:
Bei einer Satzlänge bis zu 122 Zeichen 1000 Datensätze;
bei einer Satzlänge bis zu 247 Zeichen 500 Datensätze;
Maximale Satzlänge: 247 Zeichen;
Maximale Anzahl von Feldern in einem Satz: 10;
Maximale Anzahl der Indexdateien: 3;
Maximale Länge eines Textes in KByte: 8; (8 KByte entsprechen zirka drei Schreibmaschinenseiten).
Maximale Anzahl von Texten auf einer Diskette: beliebig.

Das Programm hat wie Multidata eine einfach zu bedienende Menüsteuerung. Die Menüpunkte im einzelnen sind: Dateiaufbau, Eingabe, Suchen, Blättern, Ändern, Löschen, geordnete Listen, Wechsel von Disk, Information, Help, Parameter, Farbe, Textverarbeitung und Ready.

Um einen Programmpunkt anzuwählen, ist nur der Anfangsbuchstabe einzugeben. Da die meisten Programmpunkte sich selbst erklären, sei hier nur auf einige hingewiesen.

Der Menüpunkt Paramater steuert fast alle anderen Funktionen. Hier wird eingestellt, ob ein Datensatz ausgedruckt werden soll, ob ein automatisches Weiterblättern erfolgen soll und einiges andere mehr. So kann man durch eine bestimmte Eingabe das Programm derart einstellen, daß man ein Arbeitsprotokoll erhält.

Im Menüpunkt »Farbe« können Hintergrund- und Schriftfarbe gewählt und eingestellt werden. Hier werden alle Farben, die der Commodore 64 zur Verfügung stellt, ausgenutzt.

Help zeigt das Hauptemnü auf dem Bildschirm, denn im Programmablauf wird jeweils nur nach dem nächsten Kommando gefragt. Da das Menü nur über die jeweiligen Anfangsbuchstaben gesteuert wird, die sich aber leicht merken lassen, ist dies sicher eher ein Vorteil als ein Nachteil.

Wählt man die Textverarbeitung, erscheint folgendes Untermenü:
Holen eines Textes von Disk,
Abspeichern eines Textes auf Disk,
Schreiben eines neuen Textes,
Lesen eines Textes in Speicher,
Drucken des Textes im Speicher,
Erstellung von Serienbriefen,
Inhaltsverzeichnis der Diskette,
Wechsel der Textdiskette und
Zurück zum Hauptmenü.

Auch diese Funktionen sind wohl so deutlich, daß sie keiner Erklärung bedürfen.

Alles in allem ist der Datenmanager ein sehr gutes Datenverwaltungs- und Textverarbeitungsprogramm, von dem wir von Anfang an begeistert waren.

Fazit

Es ist wohl etwas übertrieben, wenn Data Becker schreibt, daß Datamat den Commodore 64 zum Bürorechner macht. Bei Multidata wird dies nicht behauptet, trifft aber eher zu.

Leider lassen sich beim Datamat bei falscher Bedienung Programmabstürze nicht vermeiden. Auf telefonische Anfrage teilte uns Data Becker mit, daß dies auch noch nicht behoben sei.

Dafür wird aber Anfang Mai eine Version in Maschinensprache erscheinen. Die Besitzer der jetzigen Datamat-Version können diese für zirka 25 Mark gegen die neue Version eintauschen. Weiterhin wurde uns mitgeteilt, daß im Frühsommer eine erweiterte Version des Datamat erscheint, deren Preis zwischen 250 Mark und 300 Mark liegen wird.

Multidata dagegen läßt sich recht sicher handhaben. Es ist uns im Testbetrieb nicht gelungen, das Programm abstürzen zu lassen. Die präzise Menüführung gibt dem Benutzer, im Gegenteil zu Datamat, auch nicht viel Gelegenheit, Fehler zu machen. Auch bewußt von uns provozierte Fehler führten weder zu Datenverlust noch zu Programmabsturz. Dieselbe Sicherheit gilt auch für den Datenmanager. Klare Menüführung und gute Handhabbarkeit sind auch hier vorherrschend. (H. Rieble/rg)

Immer mehr deutsche Software-Hersteller liefern mittlerweile Produkte, die auf den Commodore 64 zugeschnitten sind — so auch Maindat 64, ein Programm mit deutschsprachiger Menüführung. Ob es mit den amerikanischen Dateiverwaltungsprogrammen konkurieren kann, haben wir getestet.

Maindat 64 besteht aus den zwei Hauptteilen »Dateiarbeit« und »Datei anlegen und ändern«. Dank der weitgehend unmißverständlichen Bezeichnungen der Menüteile und dank dem guten Aufbau des Handbuchs kommt man mit dem voll menügesteuerten Programmsystem schnell zurecht. Das Handbuch ist nach didaktischen Gesichtspunkten aufgebaut: Man kann sich sofort mit der Dateiarbeit vertraut machen, ohne erst eine Datei anlegen zu müssen, da eine Datendiskette mitgeliefert wird, auf der bereits eine für Adreßverwaltung strukturierte Datei angelegt ist.

Laden und Starten

Spätestens hier erkennt man die Liebe zum Detail, mit der dieses Programm ausgearbeitet wurde. Das Bild ist grafisch sehr ansprechend und es gehört schon fast zur Imagepflege eines Programmierers, einen Zeichensatz zu verwenden, der im Design vom Standardzeichensatz abweicht. Man kann aber jederzeit auf den Commodore-Zeichensatz umschalten. Die Beendigung einer Programmaktion wird jedesmal mit einem Dreiton-Gong angezeigt. Das Programm beherrscht die deutschen Umlaute, man wird sogar gefragt, ob man Y und Z vertauscht haben will oder nicht. Der Aufforderung, Datum und genaue Zeit einzugeben, sollte man unbedingt nachkommen. Für jeden Datensatz wird nämlich ein obligatorisches Feld »Datum/Zeit« mitgeführt, welches den Zeitpunkt der Erstellung beziehungsweise der letzten Änderung enthält. Ein Hinweis für Nachtarbeiter: Während die Uhr ständig weiterläuft, wird das Datum um Mitternacht nicht fortgeschaltet. Wenn man nun noch Angaben über einen eventuell angeschlossenen Drucker gemacht hat (zu allen Fragen schlägt das Programm vernünftige Standardantworten vor), gelangt man in das Lademenü.

Lademenü

Man hat die Wahl zwischen »Dateiarbeit«, »Datei ändern«, »Neue Datei anlegen«, »Programm beenden« und sogenannten Erweiterungsmodulen (A) bis (D). Letzere sind Maschinenprogramme,

die über das Lademenü nachgeladen und gestartet werden können. In der zum Test vorliegenden Programmversion ist nur Modul (A) zur Erneuerung eines möglicherweise fehlerhaften Indexes einer Datei implementiert. Darauf wird später noch eingegangen werden. Modul (B) zum Kopieren ganzer Dateien und Modul (C) zum Erstellen einer Textverarbeitungsdatei sind angekündigt, lagen aber zum Zeitpunkt des Tests noch nicht vor. Der erfahrenere Benutzer, der selbst in Assembler programmiert, wird dankbar sein für die Hinweise im Handbuch über die Aufrufstruktur dieser Module. So kann er das ohnehin flexible Dateiverwaltungssystem für spezielle Aufgaben selbst erweitern. Es ist nun empfehlenswert, die mitgelieferte Datendiskette einzulegen und den Menüteil »Dateiarbeit« zu wählen.

Dateiarbeit

Zum Arbeiten mit bestehenden Dateien werden im Hauptmenü die üblichen Funktionen, wie »Eintragen«, »Ändern«,» Sortieren«, »Ausdrucken« und »Löschen« bereitgestellt. Dabei kann man die Datensätze, die man ändern, ausdrucken oder löschen will, nach einheitlichen Schemata suchen lassen. Diese Suchschemata kann man als durchaus brauchbar bezeichnen.

Leistungsfähige Suchschemata

So kann man zunächst natürlich alle Datensätze ansprechen. Man kann die Sätze ansprechen, bei denen irgendein Feld mit einem angegebenen Suchbegriff anfängt. Sinnvoll ist das beispielsweise in einem Literaturverzeichnis: Jeder Datensatz enthält unter anderem mehrere Felder mit Stichworten zu dem registrierten Artikel. Wenn man nun zu einem gegebenen Stichwort Artikel sucht, weiß man natürlich nicht, in welchem Feld der gesuchten Datensätze, wenn überhaupt, das gesuchte Stichwort steht. Wenn man sich nicht sicher ist, wie das gesuchte Stichwort oder ein Name geschrieben wird, (zum Beispiel Meier, Maier, Meyer) kann man nach ähnlichen Namen suchen, wobei man auch noch den gewünschten Grad der Übereinstimmung in Prozent wählen kann. Weiterhin ist natürlich auch eine selektive Suche nach Einträgen in vorgegebenen Feldern möglich; im Falle einer Adreßdatei also zum Beispiel alle Personen, deren Name mit »M« anfängt oder alle weiblichen Personen, die in einem bestimmten Ort wohnen (falls es ein Feld »männlich/weiblich« gibt). Aber auch zum Beispiel alle Personen, die zwischen 1950 und 1970 geboren sind und außerhalb von München wohnen. Mit Hilfe der Funktion »Drucken« kann man so eine Liste der Personen erstellen, die die Kriterien des selbstdefinierten Suchschlüssels erfüllen.

Zuerst muß man definieren, welche Felder man ausdrucken will (so ist zum Beispiel der Ausdruck eines Feldes »männlich/weiblich« nicht sinnvoll, da diese Information schon im Namen steckt), ob die Felder unter- oder nebeneinander, fett oder normal gedruckt werden sollen und ob die Bezeichnungen der Felder wie »Name«, »Ort«, »Straße« mit ausgegeben werden sollen. Nun kann man jeden Datensatz, der auf dem Bildschirm erscheint, mit Hilfe der RUN/STOP-Taste einzeln oder alle Datensätze, die sich mit einem der vorhin beschriebenen Suchverfahr erfassen lassen, automatisch ausdrukken. Leider wird die Geschwindigkeit eines Druckers nicht voll ausgenützt, da auf jeden Datensatz einzeln auf Diskette zugegriffen wird, und bekanntlich ist die Datenübertragung vom Laufwerk 1541 nicht sonderlich schnell. Schön wäre es zum Beispiel auch, wenn man innerhalb eines Datensatzes die Felder zwar untereinander, aber zwei oder drei solche Datensätze nebeneinander drucken könnte.

Man kann nach jedem Datensatzfeld aufsteigend oder absteigend sortieren, also zum Beispiel alphabetisch nach dem Namen oder numerisch nach Postleitzahlen oder nach dem Alter von Personen. Da beim Sortieren auf jeden Datensatz ein Diskettenzugriff gemacht wird, dauert diese Tätigkeit entsprechend lang. Es werden zirka 2,5 Sekunden pro Datensatz angegeben.

Wenn man bedenkt, daß eine Diskette gut 1000 Datensätze faßt, und wenn man diese nicht zu groß

MAIN DAT 64

anlegt, kommt man beim Sortieren schnell in den Bereich einer Stunde. Daran hat der Programmautor allerdings gedacht und bietet noch ein schnelles aber dafür weniger genaues Sortieren an. Es wird nur nach den Anfangsbuchstaben der Einträge sortiert. So kann es also schon vorkommen, daß ein Herr Abel hinter Herrn Artus zu stehen kommt. Dafür ist aber kein einziger Diskettenzugriff erforderlich, da lediglich eine Indexdatei, die sich ständig im Computer befindet, sortiert wird. Diese Indexdatei enthält die Anfangsbuchstaben aller Felder sämtlicher Datensätze und wird auch bei jedem Neueintrag und bei jeder Änderung aktualisiert. Da die aktualisierte Indexdatei erst bei Beendigung der Dateiarbeit auf Diskette zurückgeschrieben wird, sollte man seine Arbeit immer regulär über das Menü beenden und nicht etwa den Computer einfach ausschalten. Es gehen dann zwar keine Datensätze verloren, möglicherweise kann aber wegen einer fehlerhaften Indexdatei nicht mehr auf alle Datensätze zugegriffen werden.

Kleine Statistik

Mit der Funktion »Suchen & Summieren« kann man Datensätze, die einem Suchkriterium genügen, zählen und deren numerische Datenfelder einzeln aufsummieren. Enthält eine Personaldatei ein Feld »Monatsgehalt«, so könnte man leicht zum Beispiel das Durchschnittsgehalt aller Betriebsangehörigen über 40 ermitteln. Oder man denke sich eine Datei zur Verwaltung eines Schallplattenarchivs mit Einträgen über Komponist und Spieldauer der Titel. So kann man schnell in Erfahrung bringen, wieviele Stunden man ohne Wiederholung Bach hören könnte.

Fehlerhafte Benutzereingaben werden nicht akzeptiert; so ist es beispielsweise nicht möglich, in ein numerisches Datenfeld, wo eine Postleitzahl erwartet wird, etwas anderes als Ziffern einzugeben. Auch kann man einen Datensatz nicht abspeichern, bevor nicht alle Felder mit irgendwelchen Daten belegt sind. Dadurch, daß jeder Datensatz nach seiner Erstellung oder Änderung sofort auf Diskette geschrieben wird, verliert man selbst bei Stromausfall (oder wenn jemand den Stecker rauszieht) keine Daten. Schlimmstenfalls geht die Indexdatei verloren. Diese kann man aber mit dem anfangs schon erwähnten Modul (A) regenerieren. Es ist ferner möglich versehentlich gelöschte Datensätze wiederzuherstellen, wenn man sonst an der Datei noch nichts geändert hat.Es ist allerdings notwendig deren Nummer zu wissen, wenn man nicht alle Datensätze »durchblättern« will. Auch an den Datenschutz im eigentlichen Sinne des Wortes (das heißt die Daten vor dem Menschen zu schützen) wurde gedacht. So kann man erreichen, daß eine Datei nur nach Eingabe eines vierstelligen Codewortes zugänglich wird. Das wird allerdings einen erfahreneren Programmierer nicht davon abhalten, direkt auf die Datei zuzugreifen. Ich halte es daher immer noch für sicherer, wichtige Disketten wegzusperren. Es sei noch erwähnt, daß das Programm einen vernünftigen Kopierschutz besitzt, der es dem Benutzer ermöglicht, Sicherheitskopien anzufertigen. Wird das Programm von einer Sicherheitskopie geladen, so wird man vor dem Start aufgefordert, kurz die Originaldiskette, quasi als Berechtigungsnachweis, einzulegen. Auf diese Weise kann man seine Originaldiskette schonen.

Im Lademenü werden neben »Dateiarbeit« auch die Dienste »Datei ändern« und »Neue Datei anlegen« angeboten. Beide werden im Handbuch anhand von Beispielen erklärt. Man kann die Struktur einer bereits bestehenden Datei ohne Datenverluste verändern. So könnte man die Datensätze auf der mitgelieferten Datendiskette um ein Feld »männlich/weiblich« (zu Suchzwecken) oder um ein Feld »Bemerkung« erweitern. Wenn man sich über die Struktur seiner Datensätze im klaren ist, gestaltet sich das Ändern oder Neuanlegen sehr komfortabel, da man interaktiv durch die einzelnen Arbeitsschritte geführt wird. Man wird zunächst nach dem Namen der anzulegenden Datei gefragt und danach, ob diese völlig neu angelegt werden soll. In letzterem Fall wird die Diskette neu formatiert, also alle eventuell vorhandenen Daten gelöscht. Anschließend gibt man die Bezeichnungen der Datenfelder ein. Es sind bis zu 30 Felder je Datensatz mit bis zu 37 Zeichen möglich. Ein Zusatzfeld zu 14 Zeichen für Datum und Zeit des Eintrags ist obligatorisch. Man wird weiter aufgefordert, Formate für die Felder festzulegen. Hier kann man die Länge der Eingabefelder festlegen und bestimmen ob und wo zum Beispiel nur Ziffern, nur Buchstaben oder nur ja oder nein erlaubt sind, um spätere Fehleingaben weitgehend auszuschließen. In einem nächsten Arbeitsschritt kann man festlegen, daß auf bestimmten Eingabefeldern ein Ersatztext erscheinen soll, den man dann bei der Dateneingabe überschreibt oder einfach mit RETURN übernimmt. Man kann aber auch die Funktionstasten f1 bis f8 mit häufig benötigten Texten belegen. Schließlich werden in diesem Arbeitsablauf noch Standardbelegungen von Parametern definiert, die man aber jederzeit während der späteren Dateiarbeit ändern kann. Dazu gehören die Formatierung beim Drucken und die Belegung der f-Tasten. Das Handbuch erklärt alle diese Schritte ausführlich anhand eines Beispiel: »Schallplattenarchiv«.

Maindat 64 ist das erste Produkt einer angekündigten Reihe, auf die man gespannt sein kann. Vorstellbar wäre ein dazu kompatibles Textverarbeitungsprogramm oder ein Kalkulationsprogramm, das die statistische Behandlung der mit Maindat 64 erstellten Datensätze weitgehend unterstützt. Wenn ich abschließend Maindat 64 mit anderen mir bekannten Dateiverwaltungsprogrammen vergleiche, schneidet dieses Programm in den wesentlichen Beurteilungskriterien Bedienerfreundlichkeit, Geschwindigkeit, Flexibilität und Datensicherheit besser ab. Für den Preis von 128 Mark inklusive Mehrwertsteuer erhält man sicher einen reellen Gegenwert. Wenn man auf dem kurzlebigen Markt für Home-Computer-Software mit Standardprogrammen wie Dateiverwaltung Fuß fassen will, muß man sich an das Motto: »Das Bessere ist des Guten Feind« halten. Bei der Entwicklung von Maindat 64 wurde, wie mir scheint, dieser Grundsatz berücksichtigt. (Thomas Krätzig)

ISM 64 hat uns leider erst kurz vor Redaktionsschluß erreicht. Daher konnte kein ausführlicher Test mehr vorgenommen werden. Doch der erste Eindruck war hervorragend. Ein vollständiger Testbericht wird noch nachgereicht; wir wollen uns hier auf eine umfangreiche Produktvorstellung beschränken.

AUSWAHL
ISM-DEMOPROGRAMM......: D
DATEIEN ANLEGEN.......: A
BEISPIEL: BEARBEITEN.: BB
BEISPIEL: ANLEGEN....: BA
ENDE..................: E
-->

ISM 64 - Professionelle Datenverwaltung

Der ISM 64 (Index Sequential File Manager) von SM Software, ist ursprünglich für den Eigenbedarf des Herstellers entwickelt worden. Diese Vorstufe bildet die Grundlage, auf der die Anwender-Version aufgebaut wurde.

ISM 64 ist vollständig in 6502 Maschinensprache geschrieben und läuft auf den Commodore-Computern 8032, 8096 und 64 (verschiedene Objektversionen und Systemstarts). Inklusive der benötigten Pufferbereiche werden zirka 15 KByte Arbeitsspeicher im Computer belegt.

Die technischen Daten lauten:
— Variable oder feste Satzlänge von 2 bis 31875 Bytes
— Frei definierbare Aufteilung des Satzes in Feldern (1 bis 125)
— Felder fester oder variabler Länge
— Felder und Sätze können jeden Code von 0 bis 255 erhalten, also beliebig gepackte Daten
— Feldlänge maximal 255 Bytes
— maximal 40 Schlüsselfelder
— Schlüssellänge maximal 48 Bytes (46 Bytes bei mehrdeutigen Schlüsseln)
— Schlüssel sind immer sortiert (Baumstruktur)
— Der durch Löschen von Sätzen freigegebene Platz wird ohne Reorganisation wieder verwendet (aber nicht für andere Dateien freigegeben)
— Maximal 65535 Records zu maximal 254 Bytes kann eine Satzdatei umfassen (knapp 16 Millionen Bytes)
— Durch Anpassung der Recordlänge an die Satzlänge kann eine Platzoptimierung erreicht werden
— Maximal zehn ISM-Dateien werden gleichzeitig dem Anwenderprogramm zur Verfügung gestellt
— Ein einstellbarer Puffer zwischen einem KByte und zirka 16 KByte erlaubt eine Zugriffsoptimierung auf Kosten des Arbeitsspeichers
— Die Daten-Schnittstelle zum Anwenderprogramm ist ein Stringfeld, das die einzelnen Felder des Satzes enthält.
— Zur Fehlerbehandlung können zwei verschiedene, umfangreiche Statusmeldungen verwendet werden
— Als Sonderfall kann die Schlüsselverwaltung ohne Satzdatei und umgekehrt verwendet werden
— Schlüssel können nachträglich definiert und eingetragen oder gelöscht werden, ohne die anderen Schlüssel oder den Datensatz zu berühren
— Rekonstruktion der Schlüssel aufgrund der Datensätze möglich (Zusatzprogramm)
— Halbautomatische Stapelverarbeitung.

Wie man aus dieser Aufzählung ersehen kann, sind die Möglichkeiten der ISM-Datenverwaltung recht umfangreich und komfortabel. Die Hauptanwendung dieses Programms dürfte wohl im professionellen Bereich liegen, obwohl ISM 64 mit einem Preis von 140 Mark auch für den privaten Anwender erschwinglich ist. Wenn man Datenverwaltungsprogramme auf Großrechnern mit dem ISM 64 vergleicht, so bietet dieses Programm doch ähnliche Möglichkeiten. (rg)

Während die Urväter der Elektronikspiele — die Flipperautomaten — in den Spielhallen um ihre Existenz kämpfen, erleben sie auf dem heimischen Fernsehbildschirm eine wahre Renaissance. Aus dem umfangreichen Angebot haben wir »Night Mission Pinball« und »David's Midnight Magic« ausgewählt — zwei Programme auf Diskette für den Commodore 64.

Flipper auf dem Computer

Auf den ersten Blick beeindruckt »Night Mission Pinball« vor allem durch seine extrem detaillierte Grafik (Bild 1) und die ausgefeilten Toneffekte, die beim Commodore 64 gegenüber der ursprünglichen Apple II-Version erheblich verbessert und den Möglichkeiten dieses Computers angepaßt wurden. Die Thematik für die grafische und akustische Gestaltung des Flippers bilden nächtliche Bombenangriffe der Alliierten auf deutsche Städte im 2. Weltkrieg. Daher kann der Spieler, während die »Flipperkugel« die verschiedenen Ziele anschlägt, neben dem Dröhnen von Flugzeugmotoren auch das Sperrfeuer der Fliegerabwehr und Bombendetonationen hören — wobei man allerdings über den Beitrag dieser Aufmachung zum Spielwitz geteilter Meinung sein kann.

Wer Night Mission Pinball spielen will, muß — wie in Wirklichkeit — zuerst einen »Quarter« (25 Cents) einwerfen. Das geschieht jedoch nur per Tastendruck auf dem Bildschirm. Auch ein anderes Merkmal ist von Vorbildern aus der Spielhalle entlehnt. Dort kann man häufig beobachten, wie ein Spieler am Gehäuse seines Flipperautomaten rüttelt, um den Lauf der Kugel zu beeinflussen. Moderne Geräte besitzen daher in ihrem Inneren einen Sensor, der bei starken Erschütterungen des Spieltisches die gerade im Spiel befindliche Kugel disqualifiziert. Im »Night Mission Pinball« können derartige Manipulationen per Tastendruck simuliert werden, wobei auch hier zu häufiger Gebrauch dieser Funktion zur Disqualifikation — dem sogenannten »Tilt« — führt.

Bild 2. »David's Midnight Magic« ist ein Automat mit zwei »Ebenen«, die jeweils durch ein eigenes Paar Flipper kontrolliert werden. Preis: zirka 129 Mark

Bild 1. »Night Mission Pinball« besticht durch eine extrem detaillierte Grafik und ausgefeilte Toneffekte. Preis: zirka 129 Mark

»Night Mission Pinball« — ein Flipperspiel, das vom Benutzer nach eigenen Vorstellungen verändert werden kann

Um das Programm noch vielseitiger zu gestalten, wurde es mit einem speziellen Editor versehen, mit dessen Hilfe der Spieler bestimmte Parameter seines Flippers verändern kann. Dazu gehören neben der Zahl der Kugeln pro Spiel und der Punktegrenze für ein Freispiel vor allem physikalische Eigenschaften, wie die Neigung des Tisches, die Geschwindigkeit der Flipperkugel, die Empfindlichkeit des »Tilt«-Sensors und die Stärke des Impulses, der notwendig ist, um gewisse Ziele »anzuschlagen«. Auch der Zufallsgenerator, der in Computersimulationen von Flipperautomaten verwendet werden kann, um zu verhindern, daß die Kugel nach ihrem Abschuß immer auf der gleichen Bahn läuft, darf vom Benutzer programmiert werden. Da es bei manchen Einstellungen passieren kann, daß sich die Flipperkugel an irgendeiner Stelle des Spieltisches »totläuft«, hat man außerdem die Möglichkeit, alle Veränderungen vor dem Abspeichern auf Diskette vom Computer testen zu lassen. Wer überdies die Einstellungen an seinem Flipper vor den neugierigen Blicken anderer schützen will, kann dies mittels einer sechsstelligen Zahlenkombination tun.

»David's Midnight Magic« — Qualität statt Quantität

Ein ganz anderes Konzept wurde bei »David's Midnight Magic« verwirklicht. Zugunsten eines übersichtlichen Spieltisches verzichtet man hier auf jegliche optische und akustische Gags (Bild 2). Dennoch sind auch in diesem Programm die wesentlichen Funktionen eines Flippers vorhanden. Ebenso wie in »Night Mission Pinball« kann das Rütteln am Automatengehäuse per Knopfdruck simuliert werden und ein besonders flinker Spieler hat überdies die Möglichkeit, schon verloren geglaubte Kugeln mittels einer speziellen Vorrichtung — den sogenannten »Magicsave«-Magneten — zurückzugewinnen.

Dreidimensional auf zwei Ebenen

Im Gegensatz zu »Night Mission Pinball« ist »David's Midnight Magic« ein Automat mit zwei »Ebenen«, die jeweils durch ein eigenes Paar Flipper kontrolliert werden. Dieser Aufbau macht es dem Spieler leichter, mit der Kugel bestimmte Punkte auf dem Spieltisch — im wesentlichen in der oberen Hälfte — zu erreichen, während ihr Lauf bei »Night Mission Pinball« durch die Vielzahl der Hindernisse weitestgehend vom Zufall abhängt. Daher eignet sich »David's Midnight Magic« besonders gut zur Austragung spannender Flipperturniere, denn hier wird in erster Linie spielerisches Können mit hohen Punktzahlen und einem der begehrten Plätze in der Bestenliste belohnt.

(E.O. Malisch)

PUNKTE-FUELLUNG:

RAIN

In der Natur hat man noch kein Mittel gefunden, um das Waldsterben aufzuhalten. Nun setzt der saure Regen auch auf dem Commodore 64 ein. Hier wird dem Spieler aber die Möglichkeit geboten, das Waldsterben zu verhindern.

Die Methode, den sauren Regen mit einem Zeppelin aufzufangen, ist in der Natur wohl kaum anzuwenden. Auf dem Commodore 64 aber kann man so den Wald vor Schaden bewahren.

Vor Spielbeginn werden dem »Umweltschützer« die verschiedenen Arten von Wolken erklärt. So gibt es normale Regenwolken, Wolken mit Saurem Regen und die gefährlichen Gewitterwolken.

Zu Beginn des Spiels öffnet sich am rechten Bildschirmrand ein Behälter, in dem man den abgefangenen sauren Regen zu jeder Zeit abladen kann. Ist der Behälter ganz gefüllt, wird er geschlossen, von einem LKW abgeholt und entleert. Als Bonus bekommt man dann einen Zeppelin extra und ein neuer Baum wird gepflanzt.

Die Punktzahl richtet sich nach der Anzahl der abgefangenen Wolken mit saurem Regen. Die Anzahl der Füllungen im Zeppelin darf bis zum Entleeren höchstens neun betragen. Wird dieser Regen nicht aufgefangen, verdorrt der darunterliegende Baum. In dieser Stufe kann der Baum durch Andocken des Zeppelins an den Behälter oder durch Bonus wiederhergestellt werden. Wird der Baum ein zweites Mal getroffen, ist er irreparabel beschädigt und knickt um.

Menü zum Spiel »Raingame«

Ein Spiel nicht nur für Umweltschützer

Normaler Regen darf hingegen nicht abgefangen werden, da sonst die Bäume verdorren. Wird die Wolke schwarz, handelt es sich um eine Gewitterwolke. Nun sollte man möglichst schnell das Weite suchen, denn bis zu drei Blitze verlassen die Wolke in verschiedenen Richtungen. Wenn ein Blitz den Zeppelin trifft, zerschellt er mitsamt seiner Ladung. Die besondere Schwierigkeit ist die absolute Zufälligkeit der Blitzrichtung. Nur durch gute Reaktionen ist den tödlichen Blitzen zu entkommen. Obendrein flattert von Zeit zu Zeit ein Vogel

Blitze und Vögel machen einem das Leben schwer

durch die Landschaft, der es darauf abgesehen hat, den Zeppelin anzupieksen. Auch dem Vogel sollte man also ausweichen.

Das Spiel wird mit Höhe der Punktzahl immer schwieriger, da die Schnelligkeit zunimmt und der saure Regen dann zum Teil gezielt eingesetzt wird.

Raingame ist ein »umweltbewußtes« Geschicklichkeitsspiel, das durch seinen Schwierigkeitsgrad sicher nicht schnell zu beherrschen ist, und somit auch nicht schnell langweilig wird.

(rg)

Adreß- und Telefonregister

Machen Sie Schluß mit der Zettelwirtschaft! Wozu haben Sie einen Commodore 64 mit Floppy? Im Zeitalter der Elektronik verwaltet man seine Adressen mit dem Computer.

```
Speicherbelegung

memory:30719bytes
program:6208bytes
variables:0bytes
arrays:0bytes
strings:0bytes
free:24511bytes
```


Dieses Programm bietet eine komfortable Möglichkeit, Adressen und Telefonnummern auf Diskette abzuspeichern. Man kann Adressen und Telefonnummern

- eingeben
- auf dem Bildschirm ausgeben lassen
- löschen
- ändern
- suchen
- auf Diskette abspeichern
- von Diskette einlesen.

Menüsteuerung

Vom Menü aus kann man in die Programmteile »Eingabe«, »Ausgabe«, »Suchen«, »Daten speichern«, »Daten einlesen« und »Datei vorbereiten« springen (Funktionstasten). Der Programmteil »Datei vorbereiten« dient zur Eröffnung einer Datei und braucht daher nur einmal bei der ersten Benutzung des Programms ausgeführt werden. Jedesmal, nachdem Datensätze eingegeben, gelöscht oder geändert wurden, muß die Datei natürlich wieder mit »f2« abgespeichert werden.

Eingabe

Bei der Eingabe (mit »f3«) werden Name, Vorname, Geburtstag, Wohnort, Postleitzahl, Straße, Hausnummer, Telefon und Vorwahl eingetippt. Dabei kann »DEL« zum Löschen benutzt werden. Ist ein Name, Vorname etc. vollständig eingegeben, wird mit »RETURN« abgeschlossen, und der Cursor springt zum nächsten Eingabekriterium (wenn über die gesamte Linie geschrieben wird, springt der Cursor automatisch weiter).

Bei der Ausgabe kann unter »sortiert« und »physikalisch« ausgewählt werden, das heißt, die Datensätze werden entweder nach Name und Vorname sortiert ausgegeben oder in der Reihenfolge wie sie eingegeben wurden.

Beim Suchen wird zunächst abgefragt, wonach gesucht werden soll. Wird ein entsprechender Datensatz gefunden, so wird dieser dann ausgegeben. Danach kann nach dem gleichen Suchkriterium sofort weitergesucht werden. Im Such- und Ausgabemodus (hierbei nur in der sortierten Ausgabe) kann der gerade angezeigte Datensatz gelöscht oder geändert werden.

Beim Verändern der Daten muß auf jeden Fall der gesamte Teil, wie bei der Eingabe, neu überschrieben werden (GET-Schleife, kein INPUT). Soll ein Teil nicht geändert werden, so ist »f7« zu drücken und der Cursor springt weiter. Verändern lassen sich alle elf Datensatzteile, außer Name und Vorname.

Beliebig viele Daten

Die Datei ist für 100 Datensätze angelegt. Bei mehr als 100 Datensätzen ist die DIM-Anweisung für F$ in Zeile 100 entsprechend zu ändern. Die Datei selbst wird unter »ADR.-DATEI« auf Diskette abgespeichert und ist als verkettete Liste organisiert, das heißt, in jedem Datensatz ist notiert, wo sich der nächste Satz befindet. Dadurch lassen sich neu eingegebene Daten schnell einsortieren und in der Ausgabe kann zwischen »sortiert« und »physikalisch« unterschieden werden.

Alle Eingaben werden überprüft, so daß keine Fehlermeldungen (zum Beispiel »REDO FROM START«) auftreten können.

(Arne Weitzel)

Variable	Bedeutung
F$ (i,j) (i:0—100;j:0—11)	100 Datensätze (0 nicht belegt) mit jeweils 12 Daten (für mehr als 100 Datensätze ist in Zeile 100 zu ändern!)
SU$(j) (j:1—11):	Vergleichsstring beim Suchen
MO :	Modus (Eingabe, Ausgabe etc.)
L :	maximale Länge einer Eingabe
11 :	Aktuelle Länge eines Eingabestrings während einer Eingabe
SP :	Spaltenposition des Cursors
ZE :	Zeilenposition des Cursors
MA, MB :	Nummer der vorherigen Datensätze (für Einsortieren und Löschen von Datensätzen)
AN :	Anzahl der Datensätze
MO$:	Titel des Modi (»Ausgabe«, »Eingabe« etc.)
Z :	Nummer des Datensatzes

Die wichtigsten Variablen

```
Zeilen
100
110 - 140    : Dimensionierung der Felder und Ausschalten von RUN/STOP
150 - 340    : Variablenzuweisungen
360 - 430    : Menue anzeigen und Menueeingabe
450 - 510    : Eingabe von Datensaetzen
520 - 580    : Auswahl zwischen sortierter und physikalischer Ausgabe
590 - 600    : Physikalische Ausgabe
610 - 620    : Sortierte Ausgabe
630 - 660    : 'Keine weiteren Personen erfasst !'
680 - 810    : Routine zur Ausgabe eines Datensatzes
830 - 950    : GET-Schleife zur Eingabe eines Strings
970          : Zeichnen der Maske
990 -1030    : Zeichnen der Linien bei der Eingabe
1050-1090    : Einlesen der Datei von Diskette
1110-1160    : Speichern der Datei auf Diskette
1180-1310    : Einsortieren eines neu eingegebenen Datensatzes
1330-1360    : Auswahl des Suchkriteriums
1370-1390    : Eingabe des gesuchten Strings
1400-1470    : 'Eingabe zu lang !'
(1420-1450   : Suchen des Strings
1480         : nicht gefunden)
1490-1560    : Ausgabe des gefundenen Strings
1580-1620    : Weiter / Aendern / Loeschen oder Ende ?
1640-1700    : Loeschen eines Datensatzes
1720-1730    : Erzeugen des Tons
1740-1760    : Ausgabe des Datensatzes im 'AENDERN'-Modus
1770-1790    : Ueberschreiben des Strings (bzw. nicht veraendern)
1810-1840    : 'Aenderungen korrekt ?'
1860-1880    : Weiter ?
1980-2010    : Festlegen der Cursorposition und der max. Laenge eines Strings
             : Festlegen der Farben fuer Bildschirm sowie Titel fuer verschiedene
               Modi
2020-2070    : Datei vorbereiten
```

Aufschlüsselung des Adreß- und Telefonregisterprogramms nach Zeilennummer

```
10 rem*********************************
20 rem*********************************
30 rem*****      arne weitzel      ****
40 rem*****                        ****
50 rem***** ritter-von-halt-str.17 ****
60 rem*****                        ****
70 rem*****    5608 radevormwald   ****
80 rem*********************************
90 rem*********************************
100 dimf$(100,11),su$(11),na$(11):poke788,52
110 si=54272:fl=si:fh=si+1:tl=si+2:th=si+3:ww=si+4:aa=si+5:hh=si+6:vl=si+24
120 na$(1)="Name":na$(2)="Vorname":na$(3)="Geburtstag":na$(4)="Geburtsmonat"
130 na$(5)="Geburtsjahr":na$(6)="Strasse":na$(7)="Hausnummer"
140 na$(8)="Postleitzahl":na$(9)="Wohnort":na$(10)="Vorwahl":na$(11)="Telefon"
150 rem********** m e n u e *************
160 poke198,0:printchr$(142):poke53280,12:poke53281,3
170 print"[illegible]"tab(11)"[illegible]U——————————————I"
```

Listing des Basic-Programms für das Adreß- und Telefonregister

```
180 printtab(11)"|"tab(27)"|"
190 printtab(11)"J——————————K"
200 print""tab(13)"personendatei"
210 printtab(7)"J————————————————————I"
220 fori=1to13:printtab(7)"|"tab(32)"|":nexti
230 printtab(7)"J————————————————————K"
240 print""tab(9)"ausgaben              : f1"
250 printtab(9)"eingaben           : f3"
260 printtab(9)"suchen             : f5"
270 printtab(9)"daten einlesen     : f7"
280 printtab(9)"daten speichern    : f2"
290 printtab(9)"datei vorbereiten  : f4"
300 printtab(9)"ende                : f6"
310 gete$:ife$=""then310
320 if(asc(e$)<133)or(asc(e$)>139)then310
330 mo=asc(e$)-131:onmogoto170,450,360,1180,990,1050,2020,340
340 print"";:clr:restore:poke53280,14:poke53281,6:end
350 rem********** eingabe *************
360 mo$="EINGABE :":f3=151:f4=151:poke53280,15:poke53281,12:printchr$(f3)
370 print"":gosub830:printchr$(f4):gosub970
380 forj=1to11:gosub1860:sys58640
390 gosub680:f$(an+1,j)=a$:next:an=an+1
400 gosub1110:gosub1640:print:print"Weitere Eingaben (-/n)";
410 getw$:ifw$=""then410
420 ifw$="n"then160
430 goto360
440 rem*********** ausgabe *************
450 poke53280,14:poke53281,14
460 print""tab(15)"ausgabe :"
470 printtab(10)"f1 - sortiert"
480 printtab(10)"f3 - physikalisch"
490 getau$:if(au$="")or(au$<>""andau$<>"")then490
500 z=0:gosub2000
510 ifau$=""then590
520 z=z+1:ifz>anthengosub610:goto160
530 print"":iff$(z,0)="@"then520
540 gosub630:gosub1640
550 print"Weitere Ausgaben ? (-/n)";
560 getw$:ifw$=""then560
570 ifw$="n"then160
580 goto520
590 ma=z:z=val(f$(z,0)):ifz=0thengosub610:goto160
600 print"":gosub630:goto1490
610 print"Keine weiteren Personen erfasst !":gosub1640
620 fori=1to3000:next:return
630 poke53280,f1:poke53281,f2:printchr$(f3):gosub830
640 print""tab(25)"Datensatz "z;chr$(f4)
650 forj=1to11:gosub1860:sys58640
660 printf$(z,j):nextj:return
670 rem************ input ************
680 ll=0:a$="":poke204,0
690 getaa$:ifaa$=""then690
700 ifasc(aa$)<>13then730
710 ifll=0then690
720 print"_";:poke204,1:return
730 ifasc(aa$)<>20then760
740 ifll=0then690
750 print"_";:a$=left$(a$,len(a$)-1):ll=ll-1:goto690
760 ifaa$<>""then780
770 goto800
780 printaa$;:a$=a$+aa$:ll=ll+1
790 ifll<1then690
```

Listing des Basic-Programms für das Adreß- und Telefonregister (Fortsetzung)

```
800 ifpeek(207)then800
810 poke204,1:return
820 rem********** 1.maske *************
830 printchr$(14):print"█████"mo$
840 print"███";:fori=1to35:print" ";:nexti:print
850 fori=1to5:printtab(1)"█ "tab(35)" ":nexti
860 print"██";:fori=1to35:print" ";:nexti
870 print"██████████Name     :":print"████Vorname :"
880 print"████Geboren :███.██."
890 print"███Wohnhaft :"
900 print"██Str.:";
910 printtab(27)"Nr.:"
920 print"██Plz :";
930 printtab(12)"Ort :"
940 print"██Telefon :██████/"
950 return
960 rem********* 2.maske *************
970 forj=1to11:gosub1860:sys58640:fori=1tol:print"_";:nexti,j:return
980 rem********** einlesen **********
990 print"███"tab(15)"██ einlesen ":i=0:open1,8,2,"adr.-datei,s,r"
1000 forj=0to11
1010 input#1,f$(i,j)
1020 next:ifst<>64theni=i+1:goto1000
1030 close1:an=i:goto160
1040 rem*********** speichern ********
1050 print"███"tab(15)"██ speichern ":open1,8,2,"@:adr.-datei,s,w"
1060 fori=otoan
1070 forj=0to11
1080 print#1,f$(i,j)
1090 nextj,i:close1:goto160
1100 rem********* einsortieren *******
1110 ma=0:vz=val(f$(0,0)):zz=vz
1120 ifvz=0then1160
1130 mb=ma:ma=vz:vz=val(f$(ma,0))
1140 if(f$(ma,1)<f$(an,1))or((f$(ma,1)=f$(an,1))and(f$(ma,2)<f$(an,2)))then1120
1150 f$(an,0)=str$(ma):f$(mb,0)=str$(an):return
1160 f$(an,0)="0":f$(ma,0)=str$(an):return
1170 rem ********* suchen *************
1180 poke53280,14:poke53281,7:printchr$(14)
1190 print"█";:gosub1990
1200 print"███                    SUCHEN :          "
1210 print"████Wonach soll gesucht werden ?███"
1220 forj=1to11:print"██- ";na$(j)tab(20)":";
1230 ifj=10thenprint" 0":goto1260
1240 ifj=11thenprint" +":goto1260
1250 printj
1260 next
1270 gete$:ife$=""then1270
1280 if(asc(e$)<48orasc(e$)>57)and(e$<>"+")then1270
1290 ife$="0"thensu=10:goto1320
1300 ife$="+"thensu=11:goto1320
1310 su=asc(e$)-48
1320 j=su:gosub1860
1330 print"████████"na$(su)
1340 input"█████████";su$(su):iflen(su$(su))=0then1340
1350 iflen(su$(su))>1thengosub1370:goto1340
1360 goto1400
1370 gosub1640:fori=1to4
1380 print"████████████████EINGABE ZU LANG !":forj=1to200:nextj
1390 print"███                              ":forj=1to200:nextj,i:return
1400 gf=0:z=0
1410 ifval(f$(z,0))<>0thengoto1460
1420 ifgf=0thenprint"█████████Keine entsprechenden Personen erfasst !":goto1450
```

Listing des Basic-Programms für das Adreß- und Telefonregister (Fortsetzung)

```
1430 print"Weitere, fuer das Suchkriterium          zutreffende";
1440 print"Personen nicht erfasst !"
1450 gosub1640:fori=1to5000:next:goto160
1460 ma=z:z=val(f$(z,0))
1470 iff$(z,su)<>su$(su)then1410
1480 print"":gosub630:gf=1
1490 gosub1640:print"Weiter"m2$;tab(18)" - f1"
1500 print"Ende "m1$;tab(18)" - f3"
1510 print"Loeschen"tab(18)" - f5"
1520 print"Aendern"tab(18)" - f7";
1530 getw$:ifw$=""then1530
1540 la=asc(w$)-132:if(la<1)or(la>4)then1530
1550 ifmo=2thenonlagoto590,160,1590,1730
1560 onlagoto1410,160,1580,1720
1570 rem ********** loeschen **********
1580 print"LOESCHEN :":print"Datensatz "z" : ";f$(z,1)" , "f$(z,2)
1590 z1$=f$(z,0):f$(z,0)="@"
1600 f$(ma,0)=z1$
1610 forw=1to3000:next
1620 z=ma:goto1800
1630 rem************** ton ***********
1640 pokevl,15
1650 pokeaa,2*16+3
1660 poketl,0:poketh,8
1670 pokehh,15*16+2
1680 pokefh,62:pokefl,5
1690 pokeww,65:fori=1to100:next:pokeww,64
1700 return
1710 rem*********** aendern ***********
1720 mo$="AENDERN :":f1=9:f2=9:f3=155:f4=5
1730 print"":gosub970:gosub630:printtab(8)"+Keine Aenderungen - f7+"
1740 poke198,0
1750 forj=3to11:gosub1860:sys58640:gosub680:ifaa$<>""thenf$(z,j)=a$
1760 next
1770 print:gosub1640:print"Aenderungen korrekt ? (-/n)";
1780 getw$:ifw$=""then1780
1790 ifw$="n"then1730
1800 print"Weiter"m2$" (-/n)"
1810 getw$:ifw$=""then1810
1820 ifw$="n"then160
1830 ifmo=4thengosub1990:goto1410
1840 gosub2000:goto610
1850 rem********* print at ***********
1860 onjgoto1870,1880,1890,1900,1910,1920,1930,1940,1950,1960,1970
1870 l=21:sp=13:ze=5:goto1980
1880 l=21:sp=13:ze=7:goto1980
1890 l=2:sp=11:ze=11:goto1980
1900 l=2:sp=14:ze=11:goto1980
1910 l=4:sp=17:ze=11:goto1980
1920 l=18:sp=7:ze=15:goto1980
1930 l=4:sp=32:ze=15:goto1980
1940 l=4:sp=7:ze=17:goto1980
1950 l=18:sp=18:ze=17:goto1980
1960 l=5:sp=11:ze=19:goto1980
1970 l=5:sp=17:ze=19:goto1980
1980 poke214,ze:poke211,sp:return
1990 mo$="SUCHEN :":m1$="Suchen":m2$="suchen":f1=14:f2=6:f3=155:f4=158:return
2000 mo$="AUSGABE :":m1$="Ausgaben":m2$="e Ausgaben":f1=14:f2=11:f3=155
2010 f4=5:return
2020 print""tab(10)" datei vorbereiten "
2030 f$(0,0)="0":forj=1to11:f$(0,j)="-":next
2040 open1,8,2,"adr.-datei,s,w"
2050 forj=0to11
2060 print#1,f$(0,j)
2070 next:close1:goto160
```

Listing des Basic-Programms für das Adreß- und Telefonregister (Fortsetzung)

Relative Programm-Datei

Dieses Programm ist ein Beispiel, wie man mit dem VC 20 eine relative Datei erstellen und nutzen kann. Das hier vorgestellte Programm hilft den Überblick in seiner Programmsammlung zu behalten.

Eingabemaske

Durch das Sammeln und Kopieren von Programmen ergibt es sich zwangsläufig, daß gewisse Informationen wie Kopieradressen und ähnliches immer wieder gebraucht werden. Diese Angaben legt man gewöhnlich in einer Kartei ab oder schreibt sie auf einen Zettel, der wahrscheinlich bei längerem Nichtgebrauch in den Mülleimer wandert. Daraus resultiert, daß beim späteren Kopieren eines Programmes die verlorenen Informationen in mühsamer Kleinarbeit am Computer erst wieder erarbeitet werden müssen. Dies bewog mich, ein Programm zu konzipieren, mit dem die Fülle der wichtigen Daten von Programmen auf einer Diskette gespeichert und nach Bedarf wieder eingelesen werden können.

Um das Programm möglichst kurz zu halten und um keine Routinen doppelt zu schreiben, ist es in Blocks/Subroutinen geschrieben, die durch Sprungbefehle immer wieder angesprungen und genutzt werden. Deshalb besitzt das Programm eine Länge von nur 5760 Byte. Der zusätzlich benötigte Speicherplatz für die maximale Länge der Eingaben beträgt 478 Bytes. Mehr zusätzlicher Speicherplatz wird nicht benötigt, da es sich ja um eine relative Datei handelt. Das bedeutet, daß das Programm im laufenden Zustand mit insgesamt 6238 Byte Länge ohne Schwierigkeiten im VC 20 mit Erweiterungen ab 8 KByte RAM läuft. Das Programm ist ausgelegt, um die Daten von 250 Programmen aufzunehmen. Es können also genügend Daten von Programmen gespeichert werden.

Programmbeschreibung:

Nach dem Start mit RUN meldet sich das Programm-Menü auf dem Bildschirm. Von hier aus wird nach der entsprechenden Wahl in die einzelnen Routinen verzweigt. Um die Bildschirmmaske während des Programmlaufes erhalten zu können, mußte darauf verzichtet werden, weitere Menüs zu programmieren. Es sind allerdings die Funktionstasten f1/f5/f7 während des Programmlaufes belegt. Wenn man sie benötigt, wird das im oberen Bildteil optisch und über Lautsprecher auch akustisch angezeigt. Die Anzeige erfolgt in zwei Kombinationen, das heißt

a) f1 / f5 / f7
b) f1 / f7

dabei bedeutet:

f1 = Rücksprung zum Menü
f5 = Datenspeicherung
f7 = Datenänderung.

f1 — Dateneingabe

Nach dem Drücken der Taste f1 wird in die Routine zum Dateiöffnen verzweigt. Die Datei wird geöffnet (erstmaliges Öffnen der Datei nimmt einige Minuten in Anspruch) und anschließend nach der Recordnummer gefragt. Die Recordnummer ist die Nummer unter der die gesamte Eingabe auf dem Diskettenlaufwerk abgespeichert und mit der bei Abfrage wieder eingelesen wird. Nach Eingabe der Recordnummer erfolgt der Bildschirmaufbau und die Dateneingabe. Jetzt erfolgt oben beschriebenes Einblenden der Funktionstastenbelegung. Durch Drücken von f1 erfolgt ein Rücksprung ins Menü. Mit f5 wird in die Speicherroutine verzweigt und im oberen Bildteil zum Zeichen des Speicherns das Wort »SAVE« angezeigt. Anschließend wird das Hauptmenü wieder eingeblendet. Mit Druck auf f7 springt das Programm in die Eingabekorrekturroutine. Es können jetzt falsche Eingaben korrigiert werden, welche dann ohne weiteren Tastendruck durch die Speicherroutine auf der Diskette abgelegt werden.

f3 — Datenausgabe

Die Datenausgabe erfolgt über die Taste f3 vom Menü aus. Die Datei wird geöffnet, und es wird nach der einzulesenden Recordnummer gefragt. Nach Eingabe wird der entsprechende Record über die Einleseroutine (Zeilen 105 bis 123) eingelesen, die Bildschirmmaske neu erstellt und der Record über die Ausgaberoutine der Bildschirmmaske eingeschrieben. Es folgt wieder die Einblendung der belegten Funktionstasten, mit denen, wie schon beschrieben, zu verfahren ist.

f7 — Datenänderung

Durch Betätigung der Taste f7 wird in die Änderungsroutine gesprungen. Es kann hier auf weitere Programmbeschreibungen verzichtet werden, da das Programm von hier aus die bekannten und schon beschriebenen Subroutinen aufsucht, deren Handhabung und Ablauf schon ausführlich abgehandelt wurde. (Dieter Chocko)

```
0 REM******************
1 REM* PRG.DATEI 250 *
2 REM******************
3 REM*DICO-SOFT-PROG.*
4 REM*****(C)1983*****
5 POKE36879,8:POKE646,1
6 :
7 :
8 REM** MENUE **
9 PRINTCHR$(147)"          ==============          PROGRAMM-DATEI          ==========
====■"
10 PRINT">>> PROGRAMM-MENUE <<<"
11 PRINT"DATENEINGABE......<F1>"
12 PRINT"DATENAUSGABE......<F3>"
13 PRINT"DATENAENDERUNG....<F7>"
14 :
15 :
16 GETA$
17 IFA$=CHR$(133)THEN24
18 IFA$=CHR$(134)THEN105
19 IFA$=CHR$(136)THEN126
20 GOTO16
21 :
22 :
23 REM*DAT.OEFFNEN**
24 PRINT" BITTE WARTEN !! "
25 X$="                    "
26 OPEN1,8,2,"PROGRAMM-DATEI,L,"+CHR$(137)
27 OPEN2,8,15
28 PRINT#2,"P"+CHR$(2)+CHR$(250)+CHR$(0)+CHR$(1)
29 PRINT#1,CHR$(255)
30 RESTORE:INPUT"RECORDNUMMER"
31 GOSUB134:GOSUB147:GOSUB185
32 :
33 :
34 REM*DATENEINGABE*
35 INPUT"";U1$
36 U1$=LEFT$(U1$,16)
37 POKE37955,0:GOTO187
38 INPUT"";U2$
39 U2$=LEFT$(U2$,11)
40 POKE38025,0:GOTO189
41 INPUT"";U3$
42 U3$=LEFT$(U3$,11)
43 POKE38047,0:GOTO191
44 INPUT"";U4$
45 U4$=LEFT$(U4$,13)
46 POKE38112,0:GOTO194
47 INPUT"";U5$
48 U5$=LEFT$(U5$,5)
49 POKE38178,0:GOTO195
50 INPUT"";U6$
51 U6$=LEFT$(U6$,18)
52 POKE38241,0:GOTO196
53 INPUT"";U7$
54 U7$=LEFT$(U7$,18)
55 POKE38263,0:GOTO197
56 INPUT"";U8$
57 U8$=LEFT$(U8$,18)
58 POKE38285,0:GOTO198
59 INPUT"";U9$:POKE38307,0
```

Listing zu »Relative Programm-Datei«

Zeilen	Beschreibung
5	Farbcodierung
10 - 21	Menü + Tastenbelegung
24 - 32	Dateiöffnungsroutine
35 - 65	Dateieingaberoutine
68 - 84	Schriftbild F1/F5/F7 + Tastenbelegung
87 - 102	Speicherroutine
105 - 123	Einleseroutine
126 - 131	Änderungsroutine
134 - 144	Bildschirmmaske
147 - 160	Beschriftung der Bildschirmmaske
163 - 183	Korrekturroutine (Eingaben)
186 - 200	Striche für Eingabemaske
203 - 225	Ausgaberoutine + Schriftbild F1/F7 + Tastenbelegung
228 - 231	Tonroutine + Löschroutine für Schriftbilder
234 - 243	DATA-Anweisungen (Beschriftung der Bildschirmmaske/SAVE)

Programmbeschreibung anhand der Zeilennummern

```
60 U9$=LEFT$(U9$,18)
61 POKE38307,0:GOTO199
62 INPUT"[illegible]";UU$
63 UU$=LEFT$(UU$,9)
64 POKE38359,0
65 :
66 :
67 REM**T.-SCHRIFT**
68 S=4127:S1=37919:POKES,6:POKES1,7:GOSUB228
69 POKES+1,49:POKES1+1,7:GOSUB228
70 POKES+2,46:POKES1+2,7:GOSUB228
71 POKES+3,6:POKES1+3,7:GOSUB228
72 POKES+4,53:POKES1+4,7:GOSUB228
73 POKES+5,47:POKES1+5,7:GOSUB228
74 POKES+6,6:POKES1+6,7:GOSUB228
75 POKES+7,55:POKES1+7,7:GOSUB228
76 :
77 :
78 REM** TASTEN **
79 GETA$
80 IFA$=CHR$(133)THEN100
81 IFA$=CHR$(135)THENGOSUB229:GOTO87
82 IFA$=CHR$(136)THENGOSUB163:GOSUB229:GOTO87
83 GOTO79
84 :
85 :
86 REM*SPEICHERN*
87 Z$=U1$+LEFT$(X$,16-LEN(U1$))
88 Z$=Z$+U2$+LEFT$(X$,11-LEN(U2$))
89 Z$=Z$+U3$+LEFT$(X$,11-LEN(U3$))
90 Z$=Z$+U4$+LEFT$(X$,13-LEN(U4$))
91 Z$=Z$+U5$+LEFT$(X$,5-LEN(U5$))
92 Z$=Z$+U6$+LEFT$(X$,18-LEN(U6$))
93 Z$=Z$+U7$+LEFT$(X$,18-LEN(U7$))
94 Z$=Z$+U8$+LEFT$(X$,18-LEN(U8$))
95 Z$=Z$+U9$+LEFT$(X$,18-LEN(U9$))
96 Z$=Z$+UU$+LEFT$(X$,9-LEN(UU$))
97 FORB=0TO3:READK:POKE4135+B,K:POKE37927+B,7:GOSUB228:NEXTB
98 PRINT#2,"P"+CHR$(2)+CHR$(YN)+CHR$(0)+CHR$(1)
99 PRINT#1,Z$
100 CLOSE1:CLOSE2
101 GOTO9
102 :
103 :
104 REM*EINLESEN*
105 OPEN1,8,2,"PROGRAMM-DATEI,L,"+CHR$(137)
106 OPEN2,8,15
107 PRINTCHR$(147):INPUT"[illegible] RECORDNUMMER:";YN
108 PRINT#2,"P"+CHR$(2)+CHR$(YN)+CHR$(0)+CHR$(1)
109 Z$=""
110 FORI=1TO137:GET#1,O$:Z$=Z$+O$:NEXTI
111 U1$=MID$(Z$,1,16)
112 U2$=MID$(Z$,17,11)
113 U3$=MID$(Z$,28,11)
114 U4$=MID$(Z$,39,13)
115 U5$=MID$(Z$,52,5)
116 U6$=MID$(Z$,57,18)
117 U7$=MID$(Z$,75,18)
118 U8$=MID$(Z$,93,18)
119 U9$=MID$(Z$,111,18)
120 UU$=MID$(Z$,129,9)
```

Listing zu »Relative Programm-Datei (Fortsetzung)

```
121 CLOSE1:CLOSE2
122 GOSUB203
123 :
124 :
125 REM*AENDERN*
126 GOSUB105
127 GOSUB163
128 OPEN1,8,2,"PROGRAMM-DATEI,L,"+CHR$(137)
129 OPEN2,8,15
130 GOTO87
131 :
132 :
133 REM**RAHMEN**
134 PRINT"[illegible]":FORRA=4097TO4116:POKERA,64:NEXTRA
135 FORRA=4581TO4600:POKERA,64:NEXTRA
136 FORRA=4118TO4558STEP22:POKERA,93:NEXTRA
137 FORRA=4139TO4579STEP22:POKERA,93:NEXTRA
138 POKE4096,112:POKE4117,110:POKE4580,109:POKE4601,125
139 FORRA=4185TO4204:POKERA,64:NEXTRA
140 FORRA=4273TO4292:POKERA,64:NEXTRA
141 FORRA=4339TO4358:POKERA,64:NEXTRA
142 FORRA=4405TO4424:POKERA,64:NEXTRA
143 FORRA=4537TO4556:POKERA,64:NEXTRA:RETURN
144 :
145 :
146 REM**SCHRIFT**
147 PRINTCHR$(14):V1=4118
148 FORB=0TO5:READA:POKEV1+1,A:GOSUB228:FORWS=0TO5:NEXTWS:V1=V1+1:NEXTB
149 FORB=0TO14:READC:POKEV1-5+22*4,C:GOSUB228:FORWS=0TO5:NEXTWS:V1=V1+1:NEXTB
150 FORB=0TO3:READD:POKEV1+2+22*4,D:GOSUB228:FORWS=0TO5:NEXTWS:V1=V1+1:NEXTB
151 POKE4240,45
152 FORB=0TO3:READE:POKEV1-2+22*5,E:GOSUB228:FORWS=0TO5:NEXTWS:V1=V1+1:NEXTB
153 POKE4262,45
154 FORB=0TO11:READF:POKEV1-6+22*7,F:GOSUB228:FORWS=0TO5:NEXTWS:V1=V1+1:NEXTB
155 POKE4331,47
156 FORB=0TO14:READG:POKEV1-18+22*10,G:GOSUB228:FORWS=0TO5:NEXTWS:V1=V1+1:NEXTB
157 FORB=0TO4:READH:POKEV1-22+22*11,H:GOSUB228:FORWS=0TO5:NEXTWS:V1=V1+1:NEXTB
158 FORB=0TO11:READI:POKEV1-38+22*13,I:GOSUB228:FORWS=0TO5:NEXTWS:V1=V1+1:NEXTB
159 FORB=0TO7:READJ:POKEV1-50+22*19,J:GOSUB228:FORWS=0TO5:NEXTWS:V1=V1+1:NEXTB:R
ETURN
160 :
161 :
162 REM**KORREKTUR DER EINGABEN**
163 INPUT"[illegible]";U1$:POKE37955,0
164 U1$=LEFT$(U1$,16)
165 INPUT"[illegible]";U2$:POKE38025,0
166 U2$=LEFT$(U2$,11)
167 INPUT"[illegible]";U3$:POKE38047,0
168 U3$=LEFT$(U3$,11)
169 INPUT"[illegible]";U4$:POKE38112,0
170 U4$=LEFT$(U4$,13)
171 INPUT"[illegible]";U5$:POKE38178,0
172 U5$=LEFT$(U5$,5)
173 INPUT"[illegible]";U6$:POKE38241,0
174 U6$=LEFT$(U6$,18)
175 INPUT"[illegible]";U7$:POKE38263,0
176 U7$=LEFT$(U7$,18)
177 INPUT"[illegible]";U8$:POKE38285,0
178 U8$=LEFT$(U8$,18)
179 INPUT"[illegible]";U9$:POKE38307,0
180 U9$=LEFT$(U9$,18)
181 INPUT"[illegible]";UU$:POKE38359,0
182 UU$=LEFT$(UU$,9):RETURN
183 :
```

Listing zu »Relative Programm-Datei« (Fortsetzung)

Variable	Bedeutung
YN	Recordnummer
U1$- UU$	dienen zur Aufnahme der einzelnen Ein-/Ausgabezeilen
Z$	verkettet U1$— UU$zum Abspeichern und Einlesen
A - K	Read-Anweisungen
RA	Positionen für Bildschirmmaske
F	Bildschirmpositionen
FS	Farbpositionen für Striche
WS	Verzögerungsschleife
V1	Ausgangsposition zum Positionieren der READ-Anweisungen
S	Bildschirmpositionen für F1/F3/F7
S1	Farbpositionen zu S

Variablenliste

```
184 :
185 REM**R. STRICHE**
186 FORF=4165TO4180:POKEF,45:POKEF+33792,2:NEXTF:GOTO35
187 FORF=4235TO4238:POKEF,45:POKEF+33792,2:NEXTF
188 FORF=4242TO4245:POKEF,45:POKEF+33792,2:NEXTF:GOTO38
189 FORF=4257TO4260:POKEF,45:POKEF+33792,2:NEXTF
190 FORF=4264TO4267:POKEF,45:POKEF+33792,2:NEXTF:GOTO41
191 FORF=4322TO4324:POKEF,45:POKEF+33792,2:NEXTF
192 FORF=4326TO4330:POKEF,45:POKEF+33792,2:NEXTF
193 FORF=4332TO4334:POKEF,45:POKEF+33792,2:NEXTF:GOTO44
194 FORF=4388TO4392:POKEF,45:POKEF+33792,2:NEXTF:GOTO47
195 FORF=4451TO4468:POKEF,45:POKEF+33792,2:NEXTF:GOTO50
196 FORF=4473TO4490:POKEF,45:POKEF+33792,2:NEXTF:GOTO53
197 FORF=4495TO4512:POKEF,45:POKEF+33792,2:NEXTF:GOTO56
198 FORF=4517TO4534:POKEF,45:POKEF+33792,2:NEXTF:GOTO59
199 FORF=4569TO4577:POKEF,45:POKEF+33792,2:NEXTF:GOTO62:RETURN
200 :
201 :
202 REM*BILDSCHIRMAUSGABE NACH EINLESEN*
203 RESTORE:GOSUB134:GOSUB147:PRINT"[illegible]";U1$
204 PRINT"[illegible]";U2$
205 PRINT"[illegible]";U3$
206 PRINT"[illegible]";U4$
207 PRINT"[illegible]";U5$
208 PRINT"[illegible]";U6$
209 PRINT"[illegible]";U7$
210 PRINT"[illegible]";U8$
211 PRINT"[illegible]";U9$
212 PRINT"[illegible]";UU$
213 S=4127:S1=37919:POKES,6:POKES1,7:GOSUB228
214 POKES+1,49:POKES1+1,7:GOSUB228
215 POKES+2,47:POKES1+2,7:GOSUB228
216 POKES+3,6:POKES1+3,7:GOSUB228
217 POKES+4,55:POKES1+4,7:GOSUB228
218 :
219 :
220 REM** TASTEN **
221 GETA$
222 IFA$=CHR$(133)THEN9
223 IFA$=CHR$(136)THENGOSUB230:GOTO127
224 GOTO221
225 :
226 :
227 REM** TON **
228 POKE36878,15:POKE36876,220:FORWS=0TO5:NEXTWS:POKE36876,0:RETURN
229 FORX=0TO10:POKES1+X,0:NEXTX:RETURN
230 FORX=0TO5:POKES1+X,0:NEXTX:RETURN
231 :
232 :
233 REM** DATA'S **
234 DATA84,73,84,69,76,58
235 DATA75,79,80,73,69,82,65,68,82,69,83,83,69,78,58
236 DATA82,79,77,58
237 DATA82,65,77,58
238 DATA83,84,65,82,84,66,69,70,69,72,76,58
239 DATA80,82,79,71,82,65,77,77,76,65,69,78,71,69,58
240 DATA66,89,84,69,83
241 DATA66,69,77,69,82,75,85,78,71,69,78,58
242 DATA66,69,82,69,73,67,72,58
243 DATA83,65,86,69
READY.
```

Listing zu »Relative Programm-Datei« (Schluß)

Ein eigentlich unmögliches Pro

Der Drucker VC 1526 ist entgegen allen Behauptungen doch grafikfähig! Es ist nur nicht so einfach wie bei anderen Matrix-Druckern, eine hübsche Grafik zu erzeugen. Das hier vorgestellte Programm bietet die Möglichkeit, mit dem Commodore 64 erzeugte Grafiken als Hardcopy auszudrucken. Es läuft in Verbindung mit einem VC 1526-Drucker, der jedoch noch nicht mit den neuen ROMs bestückt sein darf.

Diese Hardcopys erstellt der VC 1526

Der VC 1526 bietet die Möglichkeit, ein vom Benutzer definierbares Zeichen pro Zeile auszudrucken. Zusätzlich ist ein Wagenrücklauf ohne Zeilenvorschub vorhanden. Das ist alles, reicht aber schon vollkommen aus.

Der Algorithmus besteht darin, die hochauflösende Grafik in 40 x 25 = 1000 Blöcke zu je 8 x 8 = 64 Punkte zu zerlegen und in eine für den Drucker verständliche Form umzuwandeln.

gramm

Beim Druck geschieht nun folgendes: Ein 64 Punkteblock (= 8 Byte) wird als selbstdefiniertes Zeichen an den Drucker gesendet, gedruckt und schließlich ein Wagenrücklauf ohne Zeilenvorschub erzwungen, damit das Zeichen auch auf dem Papier erscheint. Da der Druckkopf dabei an den Zeilenbeginn zurückfährt, muß er anschließend durch eine Folge von Leerschritten auf die der alten Druckposition folgende Spalte gesetzt werden. Sodann wiederholt sich der Vorgang.

Bei der Programmierung sind noch einige Features zur Geschwindigkeitsoptimierung hinzugekommen. Diese brauchen hier jedoch nicht weiter erläutert werden; wer will, kann sich aber schriftlich an mich wenden.

Ein Basic-Programm würde übrigens etwa 2 bis 5 Stunden für eine Hardcopy benötigen, das Maschinenprogramm braucht nur 2 bis 5 Minuten. Damit ist der VC 1526 genauso schnell wie ein GP 100 VC, nur mit dem Unterschied, daß er genauer ist (siehe die Hardcopy-Beispiele).

Hinweise zum Maschinenprogramm:
- Die Dateien # 125, # 126 und # 127 werden auf dem Drucker eröffnet.
- Es können nur Grafiken ausgegeben werden, die nicht vom ROM oder vom I/O-Bereich überdeckt werden, was zum Beispiel bei Simons Basic der Fall ist.
- Die Speicherzellen 07H, FBH — 10FH werden verändert.

Hinweise zur Implementierung:
- Das Basic-Programm eingeben, abspeichern und dann laufen lassen.
- Die restlichen Angaben werden vom Benutzer erfragt.
- Für verschiedene Speicherkonfigurationen können verschiedene Hardcopy-Unterprogramme angelegt werden.
- Das Hardcopy-Unterprogramm wird mit: »LOAD"HC1526/xxxx/yy", 8,1« beziehungsweise mit »LOAD "HC1526/xxxx/yy", 1,1« geladen und mit »SYS xxxxx« gestartet. Der Drucker muß dabei bereits eingeschaltet sein.

(Stefan Tramm)

```
10 REM*********************************
20 REM*  HARDCOPY FUER VC-1526  V1.0  *
30 REM*                               *
40 REM*  BY STEFAN TRAMM              *
50 REM*       BOSDORF 45              *
60 REM*       2724 HELLWEGE           *
70 REM*********************************
75 POKE 53280,6:POKE 53281,6:PRINT"♥":REM FARBEN SETZEN(BLAU,BLAU,HELLBLAU)
80 PRINT"♥▮          HARDCOPY FUER VC-1526            ■"
100 PRINT"STARTADRESSE DES ML-PROGRAMMS  49152";:INPUT"■■■■■■■";AD
110 PRINT"▮STARTADRESSE DER HIRES-GRAFIK  8192";:INPUT"■■■■■■■";HI
120 H=HI/4096:HI=INT(HI/4096)
130 IFHI<>HORHI<1ORHI>14THENPRINT"DAS KANN KEINE RICHTIGE ANGABE SEIN!":GOTO110
140 PRINT"▮TABULATOR DES DRUCKBILDES (0..40)   0";:INPUT"■■■";TA
150 IF TA>40 OR TA<0 THENPRINT"DENK DOCH MAL NACH!":GOTO140
160 PRINT"▮AUF CASSETTE(1) O. FLOPPY(8) SAVEN  8";:INPUT"■■■";GE
170 IF GE<>1 AND GE<>8 THENPRINT"TTT":GOTO160
180 PRINT"▮BITTE WARTEN, ICH HAB ZU TUN ..."
200 :
510 FOR A=0 TO 341
520 READ B:C=C+B
530 POKE AD+A,B
540 NEXT
550 IF C<>45897 THENPRINT"▮▮■PROGRAMMFEHLER!!■":END
560 :
600 REM HIRES STARTADRESSE
610 POKE AD+91,HI*16:POKE AD+297,HI*16+32
620 REM TABELLEN-ZEIGER
630 T=AD+334:TH=INT(T/256):TL=T-TH*256
640 POKE AD+115,TL:POKE AD+116,TH
650 POKE AD+129,TL:POKE AD+130,TH
660 REM TABULATOR
670 POKE AD+83,TA:POKE AD+274,TA+40:POKE AD+278,TA
680 :
700 REM ABSPEICHERN
710 N$=CHR$(34)+"HC1526/"+MID$(STR$(AD),2)+"/"+MID$(STR$(TA),2)+CHR$(34)
720 AH=INT(AD/256):AL=AD-AH*256:EH=INT((AD+343)/256):EL=AD+343-EH*256
725 PRINT"■";:REM DUNKELBLAU = HINTERGRUND
730 PRINT"P┌43,"AL":P┌44,"AH":P┌45,"EL":P┌46,"EH":CLR"
735 PRINT"▮▮S♠"N$","GE",1"
740 PRINT"▮▮▮P┌43,"PEEK(43)":P┌44,"PEEK(44)":P┌45,"PEEK(45)":P┌46,"PEEK(46);
750 PRINT":RUN770"
760 PRINT"TTTTTTTT";:POKE631,13:POKE632,13:POKE633,13:POKE198,3:REM 3*'RETURN'
765 END:REM KONTROLLE AN TASTATURPUFFER
770 PRINT"♥":REM HELLBLAU
775 PRINT"TTTTTTTTTT▮ ES IST GESCHAFFT!! ■"
780 POKE 198,0:END
790 :
999 DATA 169,127,162,004
1000 DATA160,006,032,186,255,162,000,032,189,255,032,192,255,169,126,162
1001 DATA004,160,005,032,186,255,162,000,032,189,255,032,192,255,169,125
1002 DATA162,004,160,000,032,186,255,162,000,032,189,255,032,192,255,162
1003 DATA127,032,201,255,169,020,032,210,255,169,013,032,210,255,032,204
1004 DATA255,162,125,032,201,255,169,013,032,210,255,032,204,255,169,000
1005 DATA133,254,169,000,133,251,169,032,133,252,169,000,133,253,162,007
1006 DATA169,000,157,008,001,202,016,250,162,000,160,000,177,251,061,195
1007 DATA194,240,020,169,128,133,253,138,072,073,007,170,185,195,194,029
1008 DATA008,001,157,008,001,104,170,200,192,008,048,224,160,000,232,224
1009 DATA008,048,217,138,024,101,251,133,251,144,002,230,252,165,253,208,018
1010 DATA162,125,032,201,255,169,032,032,210,255,032,204,255,024,144,088
1011 DATA048,167,162,008,202,048,068,189,008,001,221,000,001,240,245,162
1012 DATA007,189,008,001,157,000,001,202,016,247,162,125,032,201,255,169
1013 DATA141,032,210,255,166,254,240,008,169,032,032,210,255,202,016,246
1014 DATA032,204,255,162,126,032,201,255,162,007,189,008,001,032,210,255
1015 DATA202,016,247,169,013,032,210,255,032,204,255,162,125,032,201,255
1016 DATA169,254,032,210,255,032,204,255,230,254,165,254,201,040,048,017
1017 DATA169,000,133,254,162,125,032,201,255,169,013,032,210,255,032,204
1018 DATA255,165,252,201,064,048,137,162,127,032,201,255,169,034,032,210
1019 DATA255,169,013,032,210,255,032,204,255,162,125,032,195,255,162,126,032
1020 DATA195,255,162,127,032,195,255,096,128,064,032,016,008,004,002,001

READY.
```

Listing zu »Hardcopy für VC 1526«

Wo gibt's denn das? Pacman für die Grundversion des

SCHM.

Nach dem Start des Programms (siehe Listing) erscheint die Frage nach der Steuerung. Gibt man »J« ein, so spielt man mit Joystick, gibt man »N« ein, mit Tastatur. Dabei bedeuten die Tasten ».« nach links ».« nach rechts, »J« nach oben und »M« nach unten. Danach kommt die Frage nach dem Level (Schwierigkeitsgrad). Geben Sie zunächst am besten eine »1« ein. Es erscheint das Labyrinth mit Punkten, Herzen, dem »Schmatzer« und dem Monster. Die Position des Schmatzers und die des Monsters sind zufällig. Der Schmatzer hat den Mund abwechselnd offen und zu (Kaubewegung). Das Spiel beginnt bei Joysticksteuerung mit Bewegung des Knüppels in eine beliebige Richtung, bei Tastatursteuerung mit Betätigung einer beliebigen Taste. Der Schmatzer hinterläßt beim Laufen eine unsichtbare Spur. Das Monster fährt zunächst nach Zufall durch das Labyrinth; es läßt alle Punkte unberührt. Trifft das Monster auf die Spur des Schmatzers, so beginnt es, ihr — in größerer Geschwindigkeit als sich der Schmatzer bewegt — nachzulaufen was sich durch einen hohen Ton bemerkbar macht. Dabei löscht das Monster die Spur wieder. Das Monster geht nach einiger Zeit wieder von der Spur ab (Zufall). Diese Wahrscheinlichkeit wird aber von Runde zu Runde (zweistellige Zahl rechts oben, siehe Bildschirmfoto) geringer. Frißt der Schmatzer ein Herz, so wird das Monster an eine zufällige Position im Labyrinth katapultiert. Dies ist eine Möglichkeit, das Monster von der Spur des Schmatzers abzubringen.

Während des Spiels bedeutet die Zahl rechts oben die Rundenzahl, die darunter die Anzahl der Leben des Schmatzers. Schließt das Monster seine Verfolgung mit einem tieferen Ton ab, so bedeutet dies, daß es einer älteren Spur nachgelaufen war, die ein Ende gefunden hat. Wird der Schmatzer vom Monster erwischt, so wird ein Leben abgezogen, das Monster erscheint an einer zufälligen Stelle im Labyrinth und das Spiel geht durch Betätigung einer Taste beziehungsweise des Joysticks weiter. Ist die Zahl der Leben gleich Null, so ist das Spiel zu Ende. Ein neues Spiel kann mit Betätigung der Leertaste begonnen werden.

Hat man alle 265 Punkte im Labyrinth aufgefressen, wird das Labyrinth wieder mit Punkten gefüllt, und man kommt in die nächste Runde. Die Positionen des Schmatzers und des Monsters sind

Liste der Variablen von »Schmatzer«

```
HI    Hi-Score (höchste Punktzahl)
A$    Zur Eingabe des Levels und zur Wahl der Steuerung
L     Level
J
A
B     Dienen zur Tastatur- bzw. Joysticksteuerung
C
D
Z     Farbe des Monsters
I     Dient zur Mundbewegung des Schmatzers
P     Dient verschiedenen Zwecken
Q     Tonadresse (36876)
MR    Anzahl der Leben des Schmatzers
SC    Score (Punktzahl) des Spielers
PU    Anzahl der schon gefressenen Punkte
S     Nummer der Runde
T     Entscheidet, ob das Monster eine Spur hat
YM    Bewegungsrichtung des Monsters
F     Farbadresse, zur Bildschirmadresse addiert
X     Position des Schmatzers
Y     Position des Monsters
PO    Zeichen, auf dem sich das Monster befindet
XT    Zur Steuerung des Schmatzers
XM    Bewegungsrichtung des Schmatzers
P(.)  Bildschirmadressen der Herzen
```

```
5 REM SCHMATZER  VON  HANNES KALTENBACH
10 HI=0:DIMP(2):P(0)=7938:P(1)=7769:P(2)=8101:GOTO950
20 FORP=1TO500:NEXT:RETURN
30 POKE36879,25:RESTORE:FORI=7616TO7679:READP:POKEI,P:NEXT
40 Q=36876:MR=3:SC=0:PU=0:POKE36869,255:S=50
50 T=1:POKE36878,15:YM=1:I=56:F=30720:GOSUB350
60 X=INT(RND(1)*506)+7680:IFPEEK(X)<>174THEN60
70 GOSUB860:POKEX,I:POKEY,58:POKEX+F,0:POKEY+F,2
80 IFPEEK(J)=64ORPEEK(J)=126ANDPEEK(J+1)=247THEN80
90 IFX=YTHEN700
100 P=PEEK(J):IFP=ATHENXT=-1
110 IFP=BTHENXT=1
120 IFPEEK(J+1)=BTHENXT=1
130 IFP=CTHENXT=22
140 IFP=DTHENXT=-22
150 IFPEEK(X+XT)<>230THENXM=XT
160 P=PEEK(X+XM):IFP=230THEN210
170 IFP=211THEN820
180 IFP=174THENPU=PU+1:POKEQ,240:POKEQ,0:IFPU=265THEN610
190 POKEX+F,1:POKEX,XM+22:X=X+XM:POKEX,I:POKEX+F,0:IFX=YTHEN690
200 I=I+1:IFI=58THENI=56
210 IFT=1THEN260
220 POKEQ,250:POKEQ,0:POKEY,160:Y=Y+YM:YM=PEEK(Y)-22:POKEY+F,Z:POKEY,58
230 IFYM>22THEN900
240 IFINT(RND(1)*S)=0THENT=1:YM=-YM
250 GOTO90
260 P=PEEK(Y+YM):IFP=230THEN300
270 POKEY,PO:POKEY+F,6:Y=Y+YM:POKEY+F,Z:POKEY,58:PO=P
280 IFP<45THENT=0:YM=P-22:PO=160
290 GOTO90
300 P=INT(RND(1)*4):IFP=0THENYM=-1
310 IFP=1THENYM=1
320 IFP=2THENYM=22
330 IFP=3THENYM=-22
340 GOTO90
```

VC 20? Schmatzer bietet sogar noch ein bißchen mehr.

...ATZER

Sonderzeichen »Schmatzer«

265 Punkte sind bei »Schmatzer« zu sammeln, um eine Runde weiter zu kommen

```
740 IFPU=265THEN620
750 GOTO80
760 FORX=0TO19:FORY=0TO3:POKE7680+X+22*Y,160:NEXTY,X:SC=SC+PU
770 PRINT"▒▒SPIELENDE."TAB(22)"──────────"TAB(44)"PUNKTE
    :"SCTAB(59)"▒LEER-▒";
780 IFSC>HITHENHI=SC
790 PRINTTAB(66)"▒HI-SCORE:"HITAB(81)"▒TASTE"
800 IFPEEK(197)<>32THEN800
810 POKE36869,240:POKEJ+3,255:GOTO950
820 POKEX+F,1:POKEX,XM+22:X=X+XM:POKEX,I:POKEX+F,0:POKEY+F,7
830 POKEQ,230:POKE36876,0:T=1:GOSUB20:POKEY+F,2
840 POKEY,PO:GOSUB860
850 GOSUB20:GOTO90
860 P=INT(RND(1)*20)+1:Y=INT(RND(1)*21)+1:Y=7680+P+22*Y
870 PO=PEEK(Y):IFPO=230ORX=YTHEN860
880 POKEY,58:POKEY+F,2:IFPO<45THENT=0:YM=PO-22:PO=160
890 RETURN
900 POKEQ,244:POKE36876,0:T=1:PO=YM+22
910 GOTO300
920 DATA36,24,60,60,36,60,0,0,36,24,36,66,90,66,60,0,60,126,
    219,102,126,66,90
930 DATA126,36,24,52,28,36,60,0,0,36,24,52,28,32,60,0,0
940 DATA36,8,52,28,32,44,0,0,36,8,36,0,0,8,0,0,4,0,32,0,0,0
    ,0,0
950 POKE36879,27:PRINT"▒▒▒SCHMATZER":PRINT"──────────"
960 PRINT"VON HANNES KALTENBACH":PRINT"▒LEVEL? (1-8) ";
970 GETA$:L=VAL(A$):IFL<1ORL>8THEN970
980 PRINTA$:PRINT"▒JOYSTICK? (J/N) ";
990 GETA$:IFA$<>"J"ANDA$<>"N"THEN990
1000 PRINTA$:J=197:A=29:B=37:C=36:D=20
1010 IFA$="J"THENJ=37151:POKEJ+3,127:A=110:B=119:C=118:D=122
1020 Z=2:IFL>4THENL=L-4:Z=1
1030 GOTO30
```

```
350 POKE36865,156:PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒";
360 PRINT"▒.......▒............▒▒";
370 PRINT"▒.▒.▒.▒.▒.▒.▒.▒.▒.▒.▒▒";
380 PRINT"▒........▒.▒.▒.▒.▒.▒▒";
390 PRINT"▒▒▒▒▒.▒▒▒.▒...▒.▒....▒";
400 PRINT"▒▒▒...▒...▒.▒.▒...▒▒.▒";
410 PRINT"▒...▒...▒.▒.▒...▒▒▒.▒";
420 PRINT"▒.▒...▒........▒▒▒....▒";
430 PRINT"▒...▒.▒.▒.▒▒......▒▒.▒";
440 PRINT"▒.▒.▒........▒▒.▒....▒";
450 PRINT"▒...▒..▒.▒.▒.▒▒.▒....▒";
460 PRINT"▒.▒.▒▒...▒.▒..▒.▒▒▒.▒.▒▒";
470 PRINT"▒.▒...▒.▒▒.▒▒.▒.▒.▒.▒▒";
480 PRINT"▒.▒.▒.▒.....▒...▒.▒.▒▒";
490 PRINT"▒.▒.▒.▒▒▒.▒...▒.....▒▒";
500 PRINT"▒.▒.......▒.▒.▒▒.▒▒.▒▒";
510 PRINT"▒.▒▒▒▒.▒▒.▒.▒.▒.....▒▒";
520 PRINT"▒.▒.....▒.....▒.▒.▒..▒";
530 PRINT"▒.▒.▒.▒.▒.▒.▒▒▒.▒.▒▒.▒";
540 PRINT"▒.▒▒▒▒....▒.▒.....▒....▒";
550 PRINT"▒.▒▒▒▒▒.▒.▒.▒▒▒.▒.▒▒.▒";
560 PRINT"▒....................▒";
570 PRINT"▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒▒":POKE8185,230:POKE38905,6
580 POKE7700,INT(S/500)+176:POKE7701,S/50-10*PEEK(7700)+1936:IFL=1THEN600
590 FORP=0TOL-2:POKEP(P),160:NEXT
600 POKE7767,MR+176:POKE36865,38:RETURN
610 POKEX,160:X=X+XM:POKEX,I
620 SC=SC+PU:PU=0:FORX=1TO6:POKEQ,240:FORP=1TO70:NEXT:POKEQ,0:FORP=1TO30:NEXTP,X
630 GOSUB20:POKE36877,130:X=15:POKEY+F,2
640 FORI=59TO63:POKEY,I:POKE36878,X:X=X-3:FORP=1TO100:NEXTP,I:POKEY,160
650 POKE36877,0:POKE36878,15
660 PRINT"▒▒PUNKTE:"SC:IFS<>150THEN680
670 GOSUB20:FORX=1TO6:FORI=190TO240:POKEQ,I:NEXT:POKEQ,0:NEXT:MR=MR+1:GOSUB600
680 FORP=1TO3000:NEXT:S=S+50:GOTO50
690 IFPO=174THENPU=PU+1
700 POKEX,58:POKEX+F,2:FORP=245TO130STEP-1:POKE36874,P:NEXT:POKE36874,0:POKEX,56
710 FORI=59TO63:POKEX,I:FORP=1TO100:NEXTP,I:POKEX,160:I=56:T=1:YM=1
720 MR=MR-1:GOSUB600:IFMR=0THEN760
730 GOSUB20:GOSUB20:GOSUB860:POKEX,56
```

Listing des Basicprogramms »Schmatzer« für die Grundversion des VC 20

wiederum zufällig und das Spiel ist etwas schwieriger. Nach der 3. Runde bekommt man noch ein Extra-Leben, von einer Tonfolge begleitet. Gibt man am Anfang für den Level eine Zahl zwischen 2 und 5 ein, erscheinen weniger Herzen auf dem Spielfeld, wodurch sich die Abwehr des Monsters schwieriger gestaltet. Ab Level 5 wird's noch mühsamer, das Monster ist unsichtbar und nur dann zu erkennen, wenn es über Punkte läuft. Das verlangt besonders hohe Aufmerksamkeit und ist ein Nervenktizel, da man oft in Ungewißheit schwebt. Diese Levels 5 bis 8 sind eher für den geübteren Spieler gedacht.

Noch ein kleiner Tip: Um schneller durch die Kurven zu kommen, kann man auch schon vor einer Verzweigung die entsprechende Taste beziehungsweise den Joystickknüppel für die gewünschte Richtung betätigen.

(Hannes Kaltenbach)

3D-Joystick-GRAFIK

ist ein Programm für den VC 20 mit mindestens 8 KByte Speichererweiterung, das es bei völliger Ausnutzung der Bildschirmauflösung des Computers erlaubt, selbst 3 D-Grafiken mit einfachsten Mitteln zu erstellen.

Es folgt eine kleine Anleitung zum Arbeiten mit der »Joystick-Grafik«.

* Computer einschalten
* Vorprogramm zur Speicherverschiebung eingeben:
POKE44,34:POKE8704,0:
NEW
* Adresse 37151 auf Inhalt überprüfen: Falls dieser nicht 126 ist, Computer kurz aus und wieder anschalten, da sonst die Joystickfunktion beeinträchtigt ist.
* Programm laden/starten
* Nach kurzer Pause erscheint das Menü. Es gibt folgende Möglichkeiten:

Zeichnen

Es erscheint eine kurze Beschreibung der Möglichkeiten während des Zeichenvorgangs.

f1 — Das Programm kehrt zum Menü zurück (damit ist das gerade Gezeichnete nicht verloren; es kann jederzeit durch »Wiederholen« zurückgerufen werden).

f3 + f5 — Mit diesen beiden Funktionstasten ist es möglich, eine Zeichenroutine zu erstellen und diese später an jeder beliebigen Stelle des Bildschirms abzurufen.

Solche Bilder lassen sich mittels Joystick-Grafik auf komfortable Art und Weise erzeugen

4608-5115	Maschinenprogramm
5120-8159	Bildschirmaufbau/Speicherplatz für hochauflösende Grafik
8200-8699	Speicherplatz für Zeichenroutine (beliebig erweiterbar)
8704-	Basicprogramm

Speicherbelegung (dezimal)

f3 — Drücken, gewünschte Routine zeichnen, wiederum f3 drücken. An jeder beliebigen Stelle mit f5 abrufen. Die Routine »verschwindet« bei Erstellung einer neuen oder bei Unterbrechung des Programmablaufs, nicht aber bei Rückkehr zum Menü.

Funktionstasten steuern

f7 — Durch Druck auf diese Funktionstaste wird ein durch Linien oder durch den Rand begrenzter Raum ausgefüllt. Diese Funktion arbeitet von links nach rechts. Man muß sich zum Ausfüllen eines Raumes also immer ganz links in diesen Raum stellen und dann f7 betätigen, um ihn ganz auszufüllen. Diese Prozedur sowie die der Routinenerstellung erfordert einige Übung, doch lassen sich später gute Effekte erzielen.

1 + 2 — Mit diesen Tasten ist die Geschwindigkeit des Zeichenpunktes zu variieren. Um schwierigere Figuren zu malen ist es ratsam, eine langsame Geschwindigkeit zu wählen, da dann eine größere Genauigkeit erzielt werden kann. Die Tasten sind nur bei Stillstand des Punktes zu betätigen.

Gezeichnet wird mit dem Joystick. Dabei ist unbedingt darauf zu achten, daß der Rand rechts und links (durch Striche markiert) nicht überschritten wird — sonst ist es möglich, daß das Maschinenprogramm versehentlich gelöscht wird. Der Feuerknopf dient zum An- und Ausschalten der Farbe. Auf diese Einstellung ist auch beim Ausfüllen zu achten. Soll eine bereits gezeichnete Linie wieder gelöscht werden, so ist sie einfach mit dem ausgeschalteten Punkt nachzuziehen. Nun ist die Zeichen- und Bildschirmfarbe einzugeben. Bei der Bildschirmfarbe ist die Farbtabelle aus dem VC-Handbuch heranzuziehen, bei der Zeichenfarbe gelten die auf den Farben stehenden Zahlen vermindert um eins. Jetzt wird der Bildschirm umgebaut, und der Spaß kann beginnen.


```
1 POKE36879,25
3 REM******************** GRAFIK              **** BY P.BETHGE      **** 2330 E'FOER
DE    *
4 REM******************** VORPROGRAMM         **** POKE44,34        **** POKE8704,0:
NEW   *
5 PRINT"■■■■■■■■************************ GRAFIK                 ** BY FIDIBUS-SOFTWAR
E*";
6 PRINT"***********************"
9 PRINT"■■■■■■-BITTE WARTEN-"
100 REM*********************DATEN MASCH-PR.*************************
110 FORT=0TO137:READA:POKE4608+T,A:NEXTT
120 DATA 169,16,141,0,144,169,46,141,1,144,169,0,141,2,144,169,21,141,3,144
122 DATA 169,0,133,253,169,16,133,254,160,0,162,0,165,240,157,0,148,138,145,253,
24
123 DATA 152,105,19,168,201,190,208,10,160,0,230,253,165,253,201,19,240,4,232,76
,32,18
124 DATA 96
130 DATA 169,0,133,253,169,20,133,254,162,12,160,0,169,0,145,253,136,208,249,202
,240,7
131 DATA 160,0,230,254,76,76,18,169,205,141,5,144,169,19,141,2,144
133 DATA 169,255,133,253,169,19,133,254,160,160,169,128,145,253,136,208,249
134 DATA 169,63,133,253,169,31,133,254,160,160,169,1,145,253,136,208,249,96
150 FORT=0TO215:READA:POKE4750+T,A:NEXTT
160 DATA 169,128,133,249,169,160,133,253,169,25,133,254,169,0,133,163
161 DATA 169,1,133,250,160,20
164 DATA 169,127,141,34,145,173,32,145,41,128,133,251,169,255,141,34,145
165 DATA 173,31,145,41,28,24,101,251,133,251,173,31,145,41,32,133,252
166 DATA 132,165,165,249,133,166,165,253,133,167,165,254,133,168
167 DATA 165,251,201,152,208,3,32,138,19,201,24,208,6,32,138,19,32,140,19
168 DATA 201,28,208,3,32,140,19,201,20,208,6,32,140,19,32,136,19,201,148,208,3
169 DATA 32,136,19,201,132,208,6,32,136,19,32,173,19,201,140,208,3,32,173,19
170 DATA 201,136,208,6,32,173,19,32,138,19
171 DATA 165,252,201,0,208,23,165,250,201,0,208,6,169,1,133,250,208,4,169,0,133,
250
172 DATA 173,31,145,201,94,240,249
173 DATA 165,250,201,1,208,13,132,164,164,165,56,177,167,229,166,145,167,164,164
174 DATA 177,253,5,249,145,253
175 DATA 165,197,201,0,208,2,230,163,201,56,208,2,198,163,201,63,208,3,76,206,19
,96
```

Listing des Grafikprogramms

```
180 FORT=0TO69:READA:POKE5000+T,A:NEXTT
190 DATA 200,96
191 DATA 136,96
192 DATA 165,249,201,1,240,3,70,249,96,169,128,133,249,165,253,201,96,48,2,230,2
54
193 DATA 201,224,208,2,230,254,24,105,160,133,253,96
194 DATA 165,249,201,128,240,3,6,249,96,169,1,133,249,165,253,201,160,16,2,198,2
54
195 DATA 201,0,208,2,198,254,56,233,160,133,253,96
200 FORT=0TO45:READA:POKE5070+T,A:NEXTT
210 DATA 132,164,200,192,160,240,13,177,253,5,249,209,253,240,5,145,253,76,208,1
9
211 DATA 164,164,136,192,255,240,13,177,253,5,249,209,253,240,5,145,253,76,228,1
9
212 DATA 164,164,32,140,19,96
300 REM*******************BASIC-PROGRAMM***
305 POKE36879,25
310 PRINT"██████████-GRAFIK-"
320 PRINT"████1.ZEICHNEN              2.WIEDERHOLEN           3.BILD EINSPIELEN"
321 PRINT"██4.BILD AUFNEHMEN"
330 GETA$
331 IFA$="1"THEN400
332 IFA$="2"THEN550
333 IFA$="3"THEN600
334 IFA$="4"THEN500
335 GOTO330
400 REM**ZEICHNEN****
410 PRINT"██████████-GRAFIK-"
411 PRINT"██-█F1█ ZURUECK████████████-█F3█ ROUTINE ERSTELLEN█-█F5█ ROUTINE ABFRAG
EN"
412 PRINT"-█F7█ AUSFUELLEN█████████-ZEICHNEN MIT JOYSTICK-FEUERKNOPF:'AN/AUS'"
413 PRINT"-█1█ LANGSAMER███████████-█2█ SCHNELLER"
414 PRINT"-RAND NICHT UEBER       SCHREITEN!"
415 INPUT"█-ZEICHENFARBE  0████";A:POKE240,A
416 INPUT"-SCHIRMFARBE  25█████";A
433 POKE36879,A
434 SYS(4608):SYS(4672)
435 SYS(4750)
440 SYS(4772)
450 FORT=0TOPEEK(163):NEXTT
460 GETA$
461 IFA$="█"THEN480
462 IFA$="█"THEN700
463 IFA$="█"THEN720
470 GOTO440
480 PRINT"█":POKE36869,192:POKE36867,46:POKE36866,22:POKE36864,12:POKE36865,38
481 GOTO300
500 REM**BILD AUFN.**
501 PRINT"██"
510 OPEN1,1,1,"BILD"
520 FORT=5120TO8159
521 B=PEEK(T)
522 PRINT#1,B
523 NEXTT
524 CLOSE1
525 GOTO300
550 REM**WIEDERHOL***
551 INPUT"██-ZEICHENFARBE  0████";A:POKE240,A
552 INPUT"-SCHIRMFARBE  25█████";A
553 POKE36879,A
560 SYS(4608):POKE36869,205:POKE36866,19:GOTO435
600 REM**BILD EINSP.*
610 INPUT"██-ZEICHENFARBE  0████";A:POKE240,A
611 INPUT"-SCHIRMFARBE  25█████";A
620 OPEN1,1,0,"BILD"
630 POKE36879,A:SYS(4608):SYS(4672)
631 FORT=5120TO8159
632 INPUT#1,B
633 POKET,B
634 NEXTT
635 CLOSE1
640 GOTO435
650 REM**SONDERMOEGL*
700 REM
701 C=0
710 FORT=0TO499
711 SYS(4772)
712 GETA$:IFA$="█"THEN440
713 IFPEEK(251)=156THEN711
714 POKE8200+T,PEEK(251):C=C+1
715 FORR=0TOPEEK(163):NEXTR
716 NEXTT
717 GETA$:IFA$<>"█"THEN717
718 POKE8200+T,156:GOTO440
720 REM
730 FORT=1TOC
732 POKE251,PEEK(8200+T)
733 SYS(4806)
735 NEXTT
736 GOTO440
740 REM**ENDE*********************
READY.
```

Listing des Grafikprogramms (Schluß)

Wiederholen

Die Zeichen- und Bildschirmfarbe ist, wie oben beschrieben, festzulegen; daraufhin erscheint das eben Gezeichnete, und ebenfalls wieder der Zeichenpunkt in der oberen Hälfte des Bildschirms, so daß weiter gezeichnet werden kann. Damit ist es möglich, eine Zeichnung in beliebigen Farbkombinationen zu betrachten und währenddessen noch zu ändern. Der Bildschirminhalt geht auch bei Programmunterbrechung nicht verloren. Es kann einfach wieder mit RUN gestartet werden, und das Gezeichnete erscheint durch Druck auf die 2 wieder. Bei Wahl des Menüpunktes »Zeichnen« geht der alte Bildschirminhalt jedoch verloren.

Bild einspielen

Bei Druck auf die Taste 3 kann ein auf Band gespeichertes Bild in den Computer geladen werden:
— Band auf entsprechende Position bringen
— Farbkombination wählen
— »Play« drücken.

Bild aufnehmen

Natürlich kann ein eben gezeichnetes Bild auch auf Band gespeichert werden:
— Band auf leere Bandstelle spulen
— Die 4 drücken
— »Record« und »Play« drükken.

Leider dauert das Laden und Speichern der Dateien von beziehungsweise auf Kassette recht lange. Die entsprechenden Routinen können jedoch sehr einfach durch Ändern der Gerätenummer für ein Floppy-Disk-Laufwerk umgeschrieben werden.

(Philip Bethge)

Supergrafik ohne Erweiterungsmodul

Bekanntlich benötigt man für den VC 20 das Erweiterungsmodul, um hochauflösende Grafik darzustellen. Die Auflösung beträgt damit 160 mal 160 Punkte. Das abgedruckte Basicprogramm liefert aber eine Auflösung von 176 mal 184 Punkten.

Nun eine knappe Erläuterung des Programms:

Die Idee besteht darin, den Zugriff des Zeichengenerators in den freien RAM-Bereich zu legen. Immer wenn in eines der 22 x 23 = 506 Kästchen ein erster Punkt gesetzt werden soll, wird dieses Kästchen mit einem neuen Bildschirm-Code belegt. Damit können 256 der 506 Kästchen angesprochen werden, was wohl für die allermeisten Anwendungen ausreicht, insbesondere für das Erstellen von Funktionsgrafen. In Zeile 10 wird der neue Bereich für die Zeichen (5120 bis 7167) vor Basic geschützt. Die DATA-Zeilen 20 und 22 beinhalten ein kleines Maschinen-Programm, das den Bereich von 5120 bis 7167 vor jedem Programmlauf löscht (eine Basic-Routine würde dazu lästige Sekunden benötigen). In Zeile 135 wird nun der Zugriff des Zeichengenerators auf den freien RAM-Bereich bewerkstelligt.

In den Zeilen 300 bis 999 können Programmteile für die Bereitstellung der X und Y-Koordinaten eines Punktes untergebracht werden. Die Koordinate X darf Werte zwischen 0 und 175, die Koordinate Y Werte zwischen 0 und 183 annehmen. Das abgedruckte Programm generiert eine Sinuslinie.

Nach Ablauf des Programms wartet der VC 20 auf einen Tastendruck und schaltet dann in den Normal-Modus zurück.

Der eigentliche Grafikteil steht im Unterprogramm ab Zeile 4000. Zunächst werden aus X und Y die Koordinaten XS und YS eines der 506 Kästchen errechnet, danach das anzusprechende Byte BY und das Bit BI. Nach Feststellung des zu verwendenden Bildschirm-Codes in Zeile 4040 wird der Inhalt der entsprechenden Adresse im Zeichengeneratorbereich mit einer geeignet gewählten Zahl codiert. Dies führt dann zur Entstehung eines Punktes an der Stelle X, Y. Zusätzlichen Erweiterungen des Programms setzt nur der Speicher Grenzen.

(Rudolf Dörr)

```
10 POKE55,0:POKE56,20
20 DATA160,20,140,43,28,169,0,162,255,157,0,0,202,208,250,140,52,28,141,0,0,200
22 DATA192,28,208,232,96
40 FORI=7200TO7226:READZ:POKEI,Z:NEXT
80 PRINT"█SUPERGRAPHIK AUF VC 20"
85 PRINT"████████0<=X<=175":PRINT"██████0,=Y,=183"
90 PRINT"██FUNKTION IN DEN ZEILEN":PRINT"█300-999 DEFINIEREN"
92 PRINT:PRINT"COPYRIGHT R.DOERR":PRINT"GUNZENHAUSEN, DEZ.83"
93 PRINT"██TASTE DRUECKEN"
95 GETV$:IFV$=""THEN 95
110 SYS7200
132 PRINT"█"
135 POKE36869,253
200 BC=-1
300 REM WAHL V.XU.Y
302 FORX=0TO175
310 Y=91*SIN(2*π*X/175)+91
330 GOSUB4000
400 NEXT
1000 GETA$:IFA$=""THEN1000
1010 POKE36869,240
1020 END
1030 :
4000 REM UP PUNKT SETZEN
4005 Y=183-Y:X=INT(X+.5):Y=INT(Y+.5):XS=INT(X/8):YS=INT(Y/8)
4010 BY=Y-YS*8:BI=7-X+XS*8
4020 Z=7680+XS+22*YS:ZF=Z+30720
4030 IFPEEK(Z)=32THENBC+1:POKEZ,BC:POKEZF,0
4040 BS=PEEK(Z)
4050 B=5120+BS*8+BY:P=PEEK(P):P=POR2↑BI
4060 POKEB,P
4080 RETURN

READY.
```

Listing, um Grafiken mit 176 und 184 Punkten darzustellen

Fahrsimu…

Es handelt sich um ein Spiel, bei dem schnellste Reaktionen gefordert werden. Ihre Aufgabe besteht darin, ein Fahrzeug ohne »Crash« ins Ziel zu lenken. Die Steuerung erfolgt über die Tastatur oder mit einem Joystick.

Die Fahrbahnbreite und die Geschwindigkeit können in je fünf Stufen (von »Anfänger« bis »Selbstmörder«) gewählt werden.

Nach Spielende wird Ihre Gesamtpunktzahl, die von dem gewählten Schwierigkeitsgrad und der Anzahl Ihrer »Crashs« abhängt, angezeigt.

Wer nach der Lektüre dieser Beschreibung glaubt, die Sache sei einfach, dem sei wärmstens empfohlen, nur einmal die Stufe für Anfänger auszuprobieren. (Ein kleiner Trost: Das Spiel hat schon »langjährige« Autofahrer zur Verzweiflung gebracht).

Eine interessante Anwendung ist der Einsatz auf Partys (Vergleich der Fahrleistungen im nüchternen und im angetrunkenen Zustand).

Jetzt noch einiges zum Programm:

Zeile 100 bis 130: SID für »Crash«-Geräusch vorbereiten.

Zeile 200 bis 270: Anleitung schreiben

Zeile 300 bis 320: DATAs für Maschinenprogramm und Sprites einPOKEn.

```
10 REM ****************************
20 REM ***  FAHR - SIMULATOR   ***
30 REM ***  BY O. BAAKE 1/84   ***
40 REM ****************************
100 REM TON VORBEREITEN
110 SI=54272:POKESI+24,15
120 POKESI+1,248:POKESI+8,6:POKESI+15,9
130 POKESI+5,154:POKESI+12,9:POKESI+19,9
200 CL=6:GOSUB6200
210 PRINTCHR$(142):CHR$(8)
220 PRINT"■■","______________________________"
230 PRINT,"■ FAHR - SIMULATOR "
270 GOSUB5000
300 IFPEEK(12800)=120THEN330
310 C=0:FORA=12800TO13229:READB:POKEA,B:C=C+B:NEXT
315 IFC<>49567THENPRINT"PRUEFSUMMENFEHLER!":STOP
320 FORA=704TO766:READB:POKEA,B:NEXT
330 POKE251,102:POKE883,8
340 PRINT:PRINT"■ WEITER ? ";
350 GOSUB6000
360 IFX$<>"J"ANDX$<>"Y"THEN350
400 CL=6:GOSUB6200
410 PRINT"■■ *** SCHWIERIGKEITSGRAD ***
420 PRINT"■■ ■TASTE■   FUER
430 PRINT"■    ■A■  :  ANFAENGER
440 PRINT"■    ■B■  :  FORTGESCHRITTENE
450 PRINT"■    ■C■  :  LANGJAEHRIGE AUTOFAHRER
460 PRINT"■    ■D■  :  PROFIS
470 PRINT"■    ■E■  :  SELBSTMOERDER
480 PRINT"■■FAHRBAHNBREITE  ■(A-E)■ : ";
490 GOSUB6000:BR$=X$
500 BR=70-ASC(X$):IFBR<1ORBR>5THEN490
510 PRINTBR$
520 :
530 PRINT"■■GESCHWINDIGKEIT ■(A-E)■ : ";
540 GOSUB6000:GE$=X$:IFX$="@"THENGOSUB6400:GOTO530
550 GE=70-ASC(X$):IFGE<1ORGE>5THEN540
560 PRINTGE$
570 POKE252,BR*3+6:POKE253,GE*2.5
580 FORA=1TO800:NEXT
600 REM *** VORBEREITUNG
```

Listing »Fahrsimulator«

lator

FAHR - SIMULATOR
VERSUCHEN SIE IHR FAHRZEUG
UNFALLFREI BIS ZUM ZIEL ZU STEUERN.
MIT DEN BEIDEN CRSR-TASTEN KOENNEN
SIE NACH LINKS ODER RECHTS LENKEN.
CRASHES ERGEBEN MINUSPUNKTE !
WEITER ?

So werden Sie auf den »Fahrsimulator« vorbereitet

```
610 PRINT"■■"
620 CR=0:SY=PEEK(251):POKE881,3
630 SYS13094
640 REM *** COUNTDOWN
650 FORA=9TO0STEP-1
660 PRINT"■";A
670 FORB=1TO300:NEXT:NEXT
680 PRINT"■  "
700 REM *** START
710 SYS12800
720 IFPEEK(881)=255THEN1000:REM IM ZIEL
730 REM *** CRASH
740 CR=CR+1
750 PRINT"■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■■"SPC(15)"■*CRASH*■
760 SI=54272
770 POKESI+4,129:POKESI+11,129:POKESI+18,129:REM *** TON
780 FORA=1TO10:POKE53294,4:B=1↑1:POKE53294,1:NEXT
800 FORB=1TO10:PRINTSPC(15)"*CRASH*■
810 FORA=1TO120:NEXT
820 PRINTSPC(15)"■*CRASH*■
830 FORA=1TO120:NEXT:NEXT
840 PRINTSPC(15)"          ■■"
850 POKESI+4,0:POKESI+11,0:POKESI+18,0
860 IFPEEK(197)=46THENGOTO400
870 :
880 X=PEEK(870)+5+BR*1.5
890 POKE53262,X*8AND255
900 POKE53264,-(X*8>255)*128
910 FORA=1TO2000:NEXT
920 GOTO700
930 :
1000 REM *** SPIELENDE
1010 PRINT"■■","■ SPIEL BEENDET "
1020 FORA=1TO1500:NEXT
1100 REM *** AUSWERTUNG
1110 POKE53269,0
1120 PRINT"■","_______________"
1130 PRINT,"■ RESULTAT: ■"
1140 IFCRTHENCL=7:GOSUB6220
1150 IFCR>15THENPRINT"■";
1200 PRINT"■ FAHRBAHNBREITE : ";BR$
```

Zeile 330: Symbol für Fahrbahnrand und Fahrbahncharakteristik festlegen
Zeile 400 bis 580: Abfragen des Schwierigkeitsgrades und Übergabe der Werte ans Maschinenprogramm
Zeile 600 bis 630: Vorbereitung (Streckenlänge festlegen, Fahrbahn und Fahrzeug zeichnen etc.)
Zeile 640 bis 680: »Countdown«
Zeile 710: Start des Maschinen-Programms. Die Rückkehr zu Basic erfolgt erst bei einem »Crash« oder wenn das Ziel erreicht ist.
Zeile 720: Fragt Streckenzähler ab, ob Ziel schon erreicht
Zeile 730 bis 920: Crash-Routine, erhöht Crashzähler, löst Crashgeräusch aus, fragt @-Taste ab, falls diese gedrückt ist, Neustart, setzt Fahrzeug in Fahrbahnmitte
Zeile 1000 bis 1370: Auswertung, Anzeige der Gesamtpunktzahl

Wichtige Speicherzellen:
251: Bildschirmcode vom Fahrbahnrand
252: Fahrbahnbreite
253: Geschwindigkeit
880, 881: Streckenzähler low, high
883: Fahrbahncharakteristik

Bemerkung: Der Streckenverlauf ist zufällig, jedoch können verschiedene Fahrbahncharakteristika eingestellt werden. Drücken Sie dazu bei der Abfrage nach der Geschwindigkeit die @-Taste. Geben Sie danach eine Zahl zwischen 0 und 9 ein. Niedrige Zahlen bewirken wenige, große; hohe Zahlen viele, kleine Kurven.

Noch ein Hinweis zur Programmeingabe: Lassen Sie zunächst die Zeile 300 weg, bis Sie sicher sind, daß alle DATAs korrekt eingegeben wurden. Wie bei allen Programmen, die teilweise in Maschinensprache geschrieben sind, empfiehlt es sich, das Programm vor dem ersten Lauf abzuspeichern.

(Oliver Baake)

```
1210 PRINT"█ GESCHWINDIGKEIT: ";GE$
1220 PRINT"██ CRASHES          :";CR
1250 P=INT((7-GE)↑4*(7-BR)↑3/2↑CR)*100
1260 IFCR>15THENP=0
1270 IFP>5E5THENCL=5:GOSUB6200
1280 PRINT"██";P;"PUNKTE.████"
1300 REM
1310 FORA=1TO1000:NEXT
1320 IFP=0THENPRINT" TROTZDEM";
1330 PRINT" NOCH EINMAL ? ";
1340 GOSUB6000
1350 IFX$="N"THENPOKE54296,0:CL=6:GOSUB6200:PRINT:END
1360 IFX$<>"J"ANDX$<>"Y"THEN1340
1370 CL=6:GOSUB6200:GOTO400
1380 :
5000 REM *** ANLEITUNG
5060 PRINT"█ VERSUCHEN SIE IHR FAHRZEUG
5080 PRINT"█ UNFALLFREI BIS ZUM ZIEL ZU STEUERN.
5100 PRINT"██ MIT DEN BEIDEN █CRSR█-TASTEN KOENNEN
5120 PRINT"█ SIE NACH LINKS ODER RECHTS LENKEN.
5170 PRINT"██ █CRASHES ERGEBEN MINUSPUNKTE !█
5180 RETURN
6000 REM GET-ROUTINE
6010 POKE53269,0
6020 GETX$:GETX$
6030 FORA=1TO4
6040 PRINTMID$("/—\|",A,1);"█";
6050 GETX$
6060 IFX$<>""THENPRINT" █";:RETURN
6070 B=1↑1↑1
6080 NEXT:GOTO6030
6200 REM FARBE SETZEN
6210 POKE53281,CL
6220 POKE53280,CL
6230 RETURN
6400 REM FAHRBAHNCHARAKTER UND -SYMBOL
6410 PRINT"████ CHARAKTER (1-9) ? ";
6420 GOSUB6000:X=VAL(X$):IFX=0THEN6420
6430 PRINTX$:POKE883,(X-1)*10
6450 PRINT"███████ SYMBOL ? ";
6460 GOSUB6000
6470 IFX$="█"THENPRINTX$;:GOTO6460
6480 IFX$=CHR$(13)THENPRINT:RETURN
6490 PRINTX$:SY=PEEK(1274)
6500 IFSY=32ORSY=96THEN6450
6510 POKE251,SY
6520 FORA=1TO500:NEXT:RETURN
6530 :
8000 REM MASCHINENPROGRAMM
8010 DATA120,169,245,141,24,208,32,119,50,169,21,141,24,208,173,31,208,32,116
8020 DATA50,14,31,208,176,40,32,67,50,14,31,208,176,32,32,243,50,32,67,50,32
8030 DATA230,50,14,31,208,176,18,32,230,50,14,31,208,176,10,206,112,3,208,197
8040 DATA206,113,3,16,192,88,96,162,0,142,0,220,174,14,208,173,1,220
8050 DATA201,127,240,15,201,251,208,23,138,24,105,2,144,14,110,16,208
8060 DATA208,9,138,56,233,2,176,3,14,16,208,141,14,208,162,127,142,0
8070 DATA220,96,0,162,60,44,162,4,134,101,232,232,232,134,98,134,100,160
8080 DATA152,132,97,160,192,132,99,169,32,160,39,145,99,136,16,251,162,24,24
8090 DATA169,32,188,80,3,145,97,200,145,97,165,251,145,99,136,145,99,152,101
8100 DATA252,168,169,32,145,97,200,145,97,165,251,145,99,136,145,99,202,56,165
8110 DATA97,233,40,133,97,176,3,198,98,56,165,99,233,40,133,99,176,199,198
8120 DATA100,165,100,197,101,176,191,201,59,208,11,162,24,189,80,3,157,81
8130 DATA3,202,16,247,96,164,253,162,0,202,208,253,136,208,250,76,67,50,174
8140 DATA80,3,240,21,232,236,114,3,202,176,14,173,115,3,205,4,220,176,6,36
8150 DATA88,48,13,16,5,44,4,220,80,6,232,236,114,3,144,3,202,48,247,138,56
8160 DATA237,80,3,142,80,3,133,88,96,169,0,133,88,234,141,112,3,169,13,162
8170 DATA24,157,80,3,202,16,250,165,252,201,22,144,4,169,20,133,252,169,39
8180 DATA56,229,252,141,114,3,165,252,74,24,105,15,10,10,10,141,14,208,162
8190 DATA0,169,1,157,0,216,157,0,217,157,0,218,157,0,219,169,32,157,0,60,157
8200 DATA0,61,157,0,62,157,0,63,202,208,225,169,216,141,15,208,162,0,142,16
8210 DATA208,142,23,208,142,27,208,142,28,208,142,29,208,142,21,208,160,11
8220 DATA140,255,7,140,255,63,160,1,140,46,208,32,119,50,32,116,50,162,128
8230 DATA142,21,208,162,0,142,31,208,96
8240 :
8400 REM SPRITE
8410 DATA0,0,0,60,0,0,60,0,0,126,0,6,126,96,6,126,96,7,255,224,6,126,96,6,126
8420 DATA96,0,255,0,0,255,0,0,255,0,1,255,128,1,255,128,29,255,184,29,255,184
8430 DATA31,255,248,29,255,184,29,255,184,0,255,0,0,126,0

READY.

*** BEI STEUERUNG DURCH JOYSTICK FOLGENDE AENDERUNGEN ***

315  IFC<>50043THEN...
5100 PRINT"██ MIT DEM STEUERKNUEPPEL KOENNEN
8040 DATA206,113,3,16,192,88,96,162,224,142,2,220,174,14,208,173,1,220
8050 DATA201,251,240,15,201,247,208,23,138,24,105,2,144,14,110,16,208
8060 DATA208,9,138,56,233,2,176,3,14,16,208,141,14,208,162,255,142,2
```

Listing zu »Fahrsimulator« (Schluß)

Basic-Programme stutzen

Es kommt nicht selten vor, daß man Teile eines umfangreichen Basic-Programms für eine Neuentwicklung verwenden möchte, während andere Teile als überflüssig weggeschnitten werden sollen. Eine kleine Routine hilft dabei.

Exbasic Level II kennt den im VC20-Basic nicht vorhandenen Befehl DELETE, genauer: DEL-b, DELa-b, DELa-, wobei a und b explizit vorgegebene Zeilennummern sind. Das im folgenden vorgeschlagene Basic-Programm simuliert den Befehl DELa-, genauer: DEL X-, wobei der (gegebenenfalls erst vom Hauptprogramm zu berechnende) aktuelle Wert von X diejenige Basic-Zeile angibt, ab welcher (inklusiv) das im Basic-Speicher befindliche Programm gekürzt werden soll. Wir beschränken uns auf DEL X-. DEL-X und DEL X-Y lassen sich nach ähnlichem Muster aufbauen (Kopieren geeigneter Teile des Interpreters in den Kassettenpuffer und dortiges Abändern durch einige wenige POKE-Befehle), erfordern aber einen etwa doppelt so großen Aufwand.

Zunächst wird das Maschinenprogramm 50707 des Interpreters in den Kassettenpuffer kopiert. Dieses Programm 50707 berechnet die Adresse derjenigen Basic-Zeile, deren Zeilennummer in 20/21 eingegeben wird, nennen wir sie ADRX. Da 20/21 beim Abarbeiten des Programms durch die Adressenberechnung bei SYS828 in Zeile 560 gestört wird, wird die Zeilennummer-Übergabestelle in der 50707-Kopie nach 1/2 verlegt. An die 50707-Kopie im Kassettenpuffer wird eine (anschließend mit 5 POKE-Befehlen abgeänderte) Kopie eines für unsere Zwecke geeigneten Teilstücks des Maschinenunterprogramms 50756 vom Interpreters gelegt. Dieses speichert in ADRX und ADRX+1 den Wert 0 (Signal für Basic-Programmende) und setzt den Zeiger 45/46 (Variablenanfang) auf ADRX+2. Das Gesamtmaschinenprogramm im Kassettenpuffer wird per SYS828 angesprungen, und ein CLR sorgt nach Rückkehr in Basic für ein Angleichen der restlichen Basic-Zeiger.

Das Programm (das sich als ein bei Bedarf einzugebendes oder bei Neuentwicklungen schon vorsorglich vorzusehendes Hilfsprogramm (Utility) versteht) kennt keine Fehlermeldungen. Ist die Basic-Zeile X nicht vorhanden, wird der Programmrest ab der nächsten verfügbaren auf X folgenden Basic-Zeile weggeschnitten. Selbstverständlich kann sich das Hilfsprogramm auch selbst (ganz oder teilweise) stutzen. Gibt es weder die Basic-Zeile X noch eine Basic-Zeile mit höherer Nummer, so bleibt alles so, wie es ist.

```
100 X=700:GOTO500
.
.
.
500 REM:DELETEX-
510 FORI=0TO45:POKE828+I,PEEK(50707+I):NEXT
520 POKE845,2:POKE856,1
530 FORI=0TO20:POKE874+I,PEEK(50756+I):NEXT
540 POKE878,95:POKE881,95:POKE883,95:POKE890,
96:POKE895,96
550 POKE2,X/256:X=X-256*PEEK(2):POKE1,X
560 SYS828:CLR
.
.
700 REM*
701 REM*
702 REM*
710 REM*
```

Listing zu »DELETE X«

Zeile	
100	X mit Zeilennummer laden, ab welcher das Gesamtprogramm gekürzt werden soll.
510	Kopieren des Interpreter-Unterprogramms 50707 in den Kassettenpuffer.
520	Verlegen der Parameter-Eingabestellen 20/21 in der 50707-Kopie nach 1/2, da 20/21 auch von der Adressenberechnung in SYS828 (Zeile 560) benötigt wird.
530	Kopieren eines geeigneten Teilstücks des Interpreter-Unterprogramms 50756 in den Kassettenpuffer und
540	geeignetes Abändern, so daß in die von der 50707-Kopie in 95/96 gelieferte Anfangsadresse der Basic-Zeile X (oder, bei Nichtvorhandensein von X, der nächstmöglichen Basic-Zeile), sagen wir ADRX, und in ADRX+1 der Wert 0 (Basic-Ende) und in 45/46 die Adresse ADRX+2 (Variablenanfang) gelegt wird.
550	Aufspalten der in X vorgegebenen Basic-Zeilennummer in Low Byte und High Byte und Übergabe an 1/2.
560	Abarbeiten des Maschinenprogramms und anschließendes Nachstellen der restlichen Basic-Zeiger.

Programmbeschreibung zu »DELETE X«

Das Beispielsprogramm schneidet die Zeile 700 und alle darauffolgenden Zeilen weg. Bei X = 690 in Zeile 100 würde es ebenfalls Zeile 700 und alle folgenden wegschneiden. Bei X = 720 würde alles so bleiben, wie es ist. Bei X = 200 bliebe nur die Zeile 100 stehen.

Wird das Hilfsprogramm mehrfach verwendet, so reicht es zur Ablaufbeschleunigung, in Zeile 100 nach 550 zu springen. Der Teil 500 bis 540 generiert das Maschinenprogramm im Kassettenpuffer, welches natürlich solange zur Verfügung steht, wie der Kassettenpuffer nicht benötigt wird (SAVE, LOAD, VERIFY).

(Fred Behringer)

Tips und Tricks für Commodore 64

Bei diesen Tips geht es hauptsächlich um eine schöne optische Aufbereitung von Text auf dem Bildschirm.

Wer schon mal ein Programm geschrieben hat, legt sicher auch Wert auf sein Copyright. Dieses Programm lenkt die Aufmerksamkeit des Betrachters auf den Copyright-Vermerk!

Nun näheres zur Grafik: Der Computer umrahmt das Geschriebene fünf mal, so daß auf dem Bildschirm später — um das Geschriebene herum — ein Rahmen entsteht.

Nun einige Erklärungen zum Programm:
Zeile 10:
Rahmen- und Hintergrundfarbe einstellen
Zeile 90:
Bildschirm löschen
Zeile 10000:
POKE I,81 bedeutet, daß auf dem Bildschirm das Zeichen mit dem Bildschirmcode 81 (Bällchen) ausgegeben wird. Dieses Zeichen kann beliebig verändert werden (siehe im Handbuch Seite 133).
Zeile 10000:
POKE I+54272,2, wobei die 2 die Farbe angibt.
Zeile 20000:
PRINT "(BLU)(HOME)(2*CRSR DOWN()6*CRSR RIGHT)...."
Zeile 20010:
PRINT "(7*CRSR RIGHT)(CRSR DOWN)...."
Zeile 20020/30/40:
PRINT "(7*CRSR RIGHT)...."
Zeile 20050:
siehe Zeile 20010.

```
10 POKE 53280,2:POKE 53281,14
90 PRINT"▮"
100 A1=983:B1=1024
110 A2=1024:B2=2064
120 A3=2063:B3=2023
130 A4=2023:B4=983
200 FOR J=1 TO 5
210 A1=A1+41:B1=B1+39
220 FOR I=A1 TO B1:GOSUB 10000:NEXT I
230 A2=A2+39:B2=B2-41
240 FOR I=A2 TO B2 STEP 40:GOSUB 10000:NEXT I
250 A3=A3-41:B3=B3-39
260 FOR I=A3 TO B3 STEP -1:GOSUB 10000:NEXT I
270 A4=A4-39:B4=B4+41
280 FOR I=A4 TO B4 STEP -40:GOSUB 10000:NEXT I
900 NEXT J
910 GOTO 20000
10000 POKE I,81:POKE I+54272,2
10010 RETURN
20000 PRINT"▮▮▮▮▮▮▮▮▮▮COPYRIGHT BY:"
20010 PRINT"▮▮▮▮▮▮▮▮VOLKER MUECKE"
20020 PRINT"▮▮▮▮▮▮▮IM HAG 32"
20030 PRINT"▮▮▮▮▮▮▮5180 ESCHWEILER
20040 PRINT"▮▮▮▮▮▮▮TEL.:02403/4230
20050 PRINT"▮▮▮▮▮▮▮▮NOVEMBER 1983
25000 POKE 198,0:WAIT 198,1

READY.
```

Verschiedene POKES
Tastaturfunktionen:
POKE 650,255:
Ab jetzt haben alle Tasten eine REPEAT-Funktion, das heißt Dauerfunktion.
POKE 650,0:
Rücksetzung in den Normalzustand
POKE 650,127:
Dauerfunktion der Cursortasten ausschalten.
POKE 657,128:
Dieser Befehl macht die Umschaltung von der Tastatur her auf Groß- und Kleinschreibung unmöglich.
POKE 788,49:
Dieser Befehl hebt die Wirkung der STOP-Taste auf!
POKE 649,0:
Hierbei gibt der Computer die gedrückten Tasten beziehungsweise deren Zeichen nicht auf dem Bildschirm aus.
POKE 649,10:
Rücksetzung normal:

Weitere POKES:
POKE 40,1: POKE 41,8: POKE 2048,0: NEW (RETURN) LOAD (RETURN)
Mit Hilfe dieser Befehle, die im Direktmodus eingegeben werden müssen, kann man ein C-64 Programm von einer Kassette in einem, CBM-Computer laden.
POKE 792,34: POKE 793,253: wirkt wie ein RESET beim Drücken der Restore-Taste
PRINT PEEK(43)+PEEK(44)*256: Mit dieser Befehlszeile, erfährt man die aktuelle Startadresse Basic. Wenn man die Startadresse verändern will, so muß man nur die Speicherzellen 43 und 44 ändern.
10 POKE 198,0:WAIT 198,1: Warten auf eine gedrückte Taste.

Verschiedene PEEKS:
PRINT PEEK (203): Gibt an, welche Taste gedrückt wurde und zwar im Charactermodus (siehe Tabelle im Handbuch), wobei 64 bedeutet, daß keine Taste gedrückt wurde.
PRINT PEEK (57): Dieser Befehl zeigt nach einer Programmunterbrechung an, an welcher Zeilennummer das Programm gestoppt wurde.

PRINT PEEK (152): Gibt die Anzahl der geöffneten Dateien bekannt.

PRINT PEEK (182): Dieser Befehl gibt die Anzahl der Zeichenlesefehler aus.

Verschiedenes:

PRINT CHR$ (14): Umschaltung auf Groß- und Kleinschrift

PRINT CHR$ (142): Rücksetzung normal

SYS 65511: Dieser Befehl schließt alle zu diesem Zeitpunkt offenen Files.

Wörter werfen

Das nachfolgende Programm habe ich »Wörter werfen« genannt. Es schreibt zuerst in die erste Zeile den Text (immer nur 1 Zeile) und wirft ihn dann buchstabenweise den »Bildschirm hinunter«, so daß der Text dann im unteren Drittel des Bildschirms ist, der dann wiederum, diesmal aber als Ganzes nach oben wandert.

Dabei kommen verschiedene Farbkombinationen zur Geltung: Zuerst ist der Satz rot, dann blau und im letzten Statdium grün. Dieses Programm eignet sich wegen seiner schönen Trickgrafik sehr gut für Programmbeschreibungen und ähnliches.

Der Text wird ab Zeile 30 in DATAs eingetragen. Die FOR-NEXT-Schleife in Zeile 160 muß auf die Anzahl der DATAs eingestellt sein (siehe Beispiel). Dabei bitte die Dimensionierung in Zeile 30 nicht vergessen! Eingabehilfen (zu Sonderzeichen):

Zeile 100:
A$(8) = "(RVS ON) Volker Mücke (RVS OFF)"

Zeile 180:
PRINT "(CLR)(RED)"...

Zeile 240:
B$="24*(CRSR DOWN)"

Zeile 250:
PRINT "(HOME)"..."(BLU)(40*Space)(2*CRSR UP)"

Zeile 260:
PRINT "(CLR)(40*Space)"

(Volker Mücke)

```
1 REM"          COPYRIGHT BY:
2 REM
3 REM"                    VOLKER MUECKE ➛
4 :
5 :
10 BS=1024:F=55296
20 POKE 53280,1:POKE53281,1
30 DIMA$(9):A$(0)="DEMOPROGRAMM :"
40 A$(1)="DAS PROGRAMM WIRFT DIE WOERTER BUCHSTA-"
50 A$(2)="BENWEISE AUF DEN BILDSCHIRM."
60 A$(3)="TEXT KANN IN DEN ZEILEN AB 40 EINGEGE-"
70 A$(4)="BEN WERDEN.":A$(5)="ANZAHL DER ZEILEN IN ZEILE 30"
80 A$(6)="DIMENSIONIEREN !!!
90 A$(7)="C O P Y R I G H T   B Y:"
100 A$(8)=" {RVS ON}V O L K E R     M U E C K E {RVS OFF}"
110 A$(9)=" NOVEMBER 1983"
160 FORII=0TO9:A$=A$(II)
170 A=INT((40-LEN(A$))/2)
180 PRINT"{CLR}{RED}"TAB(A)A$
190 FORI=BS+ATOBS+A+LEN(A$)
200 X=PEEK(I)
210 FORN=I+40TOI+40*15STEP40
220 POKEN,X:POKEN-BS+F,6:POKEN-40,32
230 NEXTN,I
240 B$="{24*CRSR DOWN}"
245 FOR K=16 TO 1STEP-1
250 PRINT"{HOME}";LEFT$(B$,K);"{BLU}
260 FORJ=1TO2:NEXTJ:PRINT"{CRSR UP}"{42*SPACE}
280 FORJ=1TO3
290 NEXTJ,K,II
300 GOTO160
```



```
150 printchr$(14):rem" Kleinschrift
160 print"{CLR}":rem" Bildschirm loeschen
170 ifs1=1thenk1=0:goto530
180 :
190 :
200 an=7:gosub260:gosub460:gosub490:print"{CLR}"
210 an=3:gosub260:gosub460:gosub490:print"{CLR}"
220 :
230 :
240 restore:run
250 end
260 fori=1toan:reada$
270 forj=1tolen(a$):b$=mid$(a$,j,1):printb$;:gosub320
280 nextj:print:print:nexti
290 return
300 :
310 rem" Pieps-Ton
320 si=54272:fl=si:fh=si+1:tl=si+2:th=si+3:w=si+4:a=si+5:h=si+6:l=si+24
330 pokel,15:pokea,21:pokeh,0:poketh,8:poketl,0:pokefh,14:pokefl,162:pokew,17
340 forsd=1to150:nextsd
350 pokew,0
360 return
370 :
380 rem" Eintragungen in Data's
390 data" V o l k e r   M u e c k e",""
400 data" Im Hag 32",""
410 data" 5180 Eschweiler",""
420 data" Tel.:02403/4230"
430 data"{CRSR DOWN}Viel Spass wuenscht:","","  (gez.)  Volker Muecke"
440 :
450 rem" Zeischleife
460 fork=1to3000:nextk:return
470 :
480 rem" Gewehrschuss
490 si=54272:fl=si:fh=si+1:tl=si+2:th=si+3:w=si+4:a=si+5:h=si+6:l=si+24
500 forx=15to0step-1
510 pokel,x:pokea,15:pokeh,0:pokefh,40:pokefl,200:pokew,129
520 next
530 pokew,0:pokea,0
540 return
```

Trick 175

Das folgende Programm habe ich Trick 17 genannt. Es erweist sich besonders beim Gestalten von Anleitungen sehr hilfreich.

Das Programm schreibt die Wörter buchstabenweise auf den Bildschirm, das heißt, der Computer schreibt wie man liest, Buchstabe für Buchstabe und druckt nicht den ganzen Text auf einmal auf den Bildschirm.

Hier aber noch einige Erklärungen zum Umgang mit diesem Programm:

In den Zeilen ab 200, finden Sie in der Variablen AN, die Anzahl der Zeilen wieder, die Sie auf eine Bildschirmseite haben wollen. Sie können diese Variable beliebig verändern, sie sollte aber nicht größer als 10 sein, da sonst der Bildschirm »überläuft«.

In den Zeilen ab 380 definieren Sie die einzelnen Zeichen mit Hilfe von DATAs. Die Anzahl der Datas soll gleich der Summe der Werte für AN sein, das heißt in diesem Fall ist die Summe von AN=10 und wir haben auch 10 DATAs. Wollen Sie eine Leerzeile ausgeben, so können Sie einfach — wie im Programm — zwei Anführungszeichen hintereinander setzen.

Auch alle Cursorzeichen, das heißt (CRSR DOWN)...(RVS ON) etc. können benutzt werden.

Disketten

Wenn Sie über eine reichhaltige Diskettensammlung verfügen, ist sicherlich schon der Wunsch aufgetaucht, den Namen und/oder die ID einer bestimmten Diskette zu ändern, ohne dabei den Disketteninhalt zu zerstören. Wir zeigen Ihnen, wie es geht.

Mit dem DOS-Befehl »RENAME« ist es möglich, bestehenden Dateien einen neuen Namen zu geben. Dieser Befehl wirkt leider nur auf Dateiebene und nicht auch auf Diskettenebene, das heißt, es ist nicht möglich, den Diskettennamen und/oder die ID zu verändern. Dazu existiert der DOS-Befehl »NEW«, der den manchmal unerwünschten Nebeneffekt hat, daß sämtliche Programme und Dateien auf der Diskette gelöscht werden. Diesen Nachteil behebt das nachfolgend beschriebene Programm (siehe Listing).

Mit diesem Programm ist es möglich, den Namen und/oder die ID einer Diskette zu ändern, ohne daß der Disketteninhalt gelöscht wird. Es läuft auf jedem Commodore-Computer mit minimal 40 Zeichen pro Zeile. Durch kleine Änderungen bei der Bildschirmausgabe kann es auch auf einem VC 20 verwendet werden. Es können alle Floppys mit DOS 2A (also 1540, 1541, 4031, 4040) verwendet werden, wobei darauf zu achten ist, daß sich bei Doppelfloppys die zu ändernde Diskette in Drive 0 befindet.

Programmbedienung

Nachdem das Programm von Diskette oder Kassette geladen und mit RUN gestartet wurde, erscheint zuerst die Frage, ob die zu ändernde Diskette eingelegt ist. Ist dies nicht der Fall, kann das Programm gegebenenfalls durch Eingabe von »N« unterbrochen und durch Drücken der RETURN-Taste fortgesetzt werden. Als nächstes erscheint der Name und die ID der eingelegten Diskette auf dem Bildschirm.

Beantwortet man die Frage, ob der Name geändert werden soll, mit »J«, besteht die Möglichkeit, den neuen Diskettennamen einzugeben. Falls der eingegebene Name länger als 16 Zeichen ist, werden automatisch nur die ersten 16 verwendet. Gleiches gilt auch für die Eingabe der ID, deren maximale Länge 2 Zeichen beträgt.

Danach wird die Diskette geändert. Das Ende dieses Vorganges wird durch die Ausgabe von »DISKETTE GEÄNDERT — = — PROGRAMMENDE« angezeigt. Die Diskette ist nun mit dem neuen Namen versehen. Doch wie kam es dazu? Um diese Frage beantworten zu können, sollte man sich die Datenstruktur auf der Diskette etwas näher betrachten.

Die Diskette ist in 35 konzentrische Spuren aufgeteilt, die aus mehreren Sektoren bestehen. Jeder Sektor ist 256 Byte lang. Damit sich der Computer auch später noch zurechtfindet, wird ein Inhaltsverzeichnis angelegt (das sogenannte DIRECTORY). In diesem Inhaltsverzeichnis befindet sich eine Liste der belegten und freien Sektoren, der Diskettenname, die ID und eine Liste der auf der Diskette gespeicherten Programme und Dateien.

Der hier interessierende Teil — Diskettenname und ID — ist auf Spur 18, Sektor 0 gespeichert. Der Name belegt in diesem Sektor die Bytes 144 bis 161, die ID 162 und 163. Nun stellt sich die Frage, wie diese Byte geändert werden können. Zu diesem Zwecke stellt das DOS sogenannte Direkt-Zugriffsbefehle zur Verfügung. Von besonderer Bedeutung sind hier:

BLOCK-READ (Abkürzung: U1): liest den Inhalt eines Sektors in den Pufferspeicher der Floppy.

BLOCK-WRITE (Abkürzung: U2): schreibt einen Datenblock aus dem Pufferspeicher in den angegebenen Sektor.

BUFFER-POINTER (Abkürzung: B-P): setzt einen Zeiger auf jenes Byte im Pufferspeicher, auf das zugegriffen werden soll.

Zum Ändern des Namens geschieht nun folgendes:

1. Das Directory wird in den Pufferspeicher gelesen.
2. Der alte Name wird durch den neuen ersetzt.
3. Die ID wird geändert.
4. Das geänderte Directory wird aus dem Pufferspeicher auf die Diskette übertragen.

Zur Übermittlung der Befehle muß zuerst der Befehlskanal mit OPEN 1,8,15 geöffnet werden. Bevor nun etwas in den Pufferspeicher geladen werden kann, muß ein Kanal für diesen Puffer angelegt werden. Dies geschieht mit OPEN 8,8,8,"#".

Das Lesen des Directory erfolgt mit PRINT #1, "U1:";8;0;18;0. Dadurch wird ein Datenblock von Laufwerk 0, Spur 18, Sektor 0 in dem Kanal 8 zugeordneten Puffer gelesen. Nun muß der Zeiger auf das erste Byte des Diskettennamens gesetzt werden: PRINT#1,»B-P«;8;144. Jetzt kann der alte Name überschrieben werden. Dies geschieht mit der PRINT#8,DN$, wobei in DN$ der neue Diskettenname gespeichert ist. DN$ muß genau 16 Byte lang sein. Das gleiche muß zum Ändern der ID gemacht werden (Zeile 1260, 1270).

Der geänderte Name steht nun im Pufferspeicher, er muß jetzt nur noch zurück auf die Diskette gebracht werden, dies geschieht durch PRINT#,"U2:";8;0;18;0 (schreibt den Inhalt des Pufferspeichers, der Kanal 8 zugeordnet ist, auf Spur 18, Sektor 0 der in Laufwerk 0 befindlichen Diskette). Die Diskette ist somit geändert.

(Uwe Repplinger)

Zauberei

Alter Name
Neuer Name

```
1 REM" ************************************************************************
2 REM" ++++++                  DISKETTENNAME ÄNDERN                      ++++++
3 REM" ++++++                  --------------------                      ++++++
4 REM" ************************************************************************
5 REM" ++++++                 COPYRIGHT (C) 1983  BY                     ++++++
6 REM" ++++++       UWE REPPLINGER, KAISERSTR. 30, 1000 BERLIN 42        ++++++
7 REM" ************************************************************************
8 :
9 :
1000 PRINT "<CLR><REVON>   =-=   DISKETTENNAME ÄNDERN   =-=   "
1010 PRINT "<REVON>          (C) BY  UWE REPPLINGER         "
1020 INPUT "<DOWN>RICHTIGE DISKETTE EINGELEGT   J<3*LEFT>";W$
1030 IF W$<>"J" THEN PRINT "<DOWN>CONT<3*UP>" : END
1040 PRINT "<DOWN>DISKNAME:";SPC(19);"DISK ID:" : PRINT TAB(10);"<UP><REVON>";
1050 OPEN 1,8,15 : PRINT#1,"I0"
1060 OPEN 8,8,8,"#"
1070 PRINT#1,"U1:";8;0;18;0
1080 PRINT#1,"B-P:";8;144
1090 FOR I=1 TO 16
1100 GET#8,CH$ : PRINT CH$;
1110 NEXT I
1120 PRINT#1"B-P:";8;162
1130 GET#8,C1$,C2$
1140 PRINT SPC(11);C1$;C2$
1150 INPUT "<2*DOWN>SOLL DER NAME GEÄNDERT WERDEN   J<3*LEFT>";W$ : G%=0
1160 IF W$<>"J" THEN 1210
1170 INPUT "<DOWN>NEUER DISKETTENNAME ";DN$
1180 DN$=LEFT$(DN$+"                ",16)
1190 PRINT#1,"B-P:";8;144
1200 PRINT#8,DN$ : G%=1
1210 INPUT "<2*DOWN>SOLL DIE DISK ID GEÄNDERT WERDEN   N<3*LEFT>";W$
1220 IF W$<>"J" THEN 1270
1230 INPUT "<DOWN>NEUE DISK ID ";ID$
1240 ID$=LEFT$(ID$+"  ",2)
1250 PRINT#1,"B-P";8;162
1260 PRINT#8,ID$ : R%=1
1270 IF G%=1 THEN PRINT#1,"U2:";8;0;18;0
1280 PRINT#1;"I0" : CLOSE 8 : CLOSE 1
1290 PRINT "<2*DOWN><REVON>"; : IF G%=1 THEN PRINT " DISKETTE GEÄNDERT  -=-";
1300 PRINT "  PROGRAMMENDE  "
1310 END
```


Programm, mit dem der Name und/oder die ID einer Diskette geändert werden kann

<CLR>	= CLEAR HOME
<REVON>	= CTRL REVERS ON
<DOWN>	= CURSOR NACH UNTEN
<UP>	= CURSOR NACH OBEN
<LEFT>	= CURSOR NACH LINKS

Steuerzeichen

Programmzeile	Erläuterung
1050	Befehlskanal eröffnen und Disk initialisieren
1060	öffnen des Datenkanals
1070	Directory in den Pufferspeicher lesen
1080	Zeiger auf Diskettennamen setzen
1090 – 1110	Diskettennamen lesen und ausgeben
1120	Zeiger auf ID setzen
1130 – 1140	ID lesen und ausgeben
1180	Diskettenname auf 16 Zeichen formatieren
1190	Zeiger auf Diskettennamen setzen
1200	Neuen Diskettennamen in den Puffer schreiben
1240	ID auf 2 Zeichen formatieren
1250 – 1260	Zeiger auf ID setzen und neue ID in den Puffer schreiben
1270	Geändertes Directory auf Diskette speichern
1280	Diskette initialisieren, Daten- und Befehlskanal schließen

Programmbeschreibung anhand der Zeilennummern

Name	Belegung
W$	Benutzereingabe
CH$	Alter Diskettenname
C1$	Erstes Zeichen der alten ID
C2$	Zweites Zeichen der alten ID
DN$	Neuer Diskettenname
ID$	Neue ID
G%	G%=1 : Diskette wurde geändert G%=0 : Diskette wurde nicht geändert

Variablendefinition

Schat

Der Name des Programms scheint zu täuschen. Denn die Schätze, die Sie suchen sollen, sind nicht irgendwo versteckt, im Gegenteil, sie liegen offen vor Ihnen. Nur beim Aufsammeln dieser wertvollen Gegenstände werden Sie von Monstern in nicht unerheblicher Art und Weise behindert. Schlagen Sie den Monstern ein Schnip Sie Reaktionsvermögen und Übersicht, vor al

Dieses Spiel für den Commodore 64 zeichnet sich durch seine Variationsvielfalt aus. Wie man aus dem Bildschirmfoto erkennen kann, befinden sich die Schätze auf mehreren Etagen verteilt, die wiederum untereinander mit Leitern verbunden sind. Die Verteilung der Schätze ist völlig zufällig, es kann daher auch vorkommen, daß in einem Stockwerk kein Schatz plaziert ist. Vor Beginn des Spiels können Sie die Laufgeschwindigkeit des Männchens — und natürlich die damit korrelierte Bewegungsvitalität der Monster — in Stufen von Eins bis Zehn einstellen. Je höher der Schwierigkeitsgrad, desto mehr Punkte können Sie erzielen. Aber Vorsicht, es ist noch kein Meister vom Himmel gefallen. Die Spielstärke 10 dürfte für so manchen zur Erkenntnis der Selbstüberschätzung führen.

Haben Sie die erste Spielrunde überstanden, was nicht allzu schwierig ist, da Sie nur von einem Monster belästigt werden, so wird das gesamte Spielfeld neu aufgebaut. Das heißt die Leitern werden in einer völlig anderen Anordnung mit unterschiedlicher Länge positioniert. Auch das Monster hat sich vermehrt, nun wollen zwei Ungeheuer Ihren Erfolg verhindern. Bei jeder neuen Spielrunde kommt ein weiteres Monster hinzu. Die Sache beginnt knifflig zu werden. Jetzt sind Fingerspitzengefühl und Übersicht gefragt. Insgesamt haben Sie vier Schatzsucher zur Verfügung, um bis ans »Ende« der Strapazen zu gelangen.

Sie können Ihren Schatzsucher entweder über die Tastatur oder mit einem Joystick steuern. Und nun viel Spaß bei der Jagd auf den High Score. (G. Klauser)

1. Programmaufbau:

Zeile 20:	Basic-RAM nach oben begrenzen
Zeile 80:	Spielanleitung aufrufen Maschinenprogramm laden
Zeile 110-190:	Spriteform lesen
Zeile 210-220:	Video Bank 2 ($8000-$BFFF) wählen, damit genügend Platz zum Speichern der Sprites vorhanden ist.
Zeile 230	Form des Schatzes
Zeile 240-400:	Sprites initialisieren, Spielfeld aufbauen
Zeile 410-480:	Bewegen und Abfragen der Sprites/ Tastatur
Zeile 490-640:	Initialisieren der Geister
Zeile 650-770:	Laden des Maschinenprogramms
Zeile 780-1050:	Subroutine alle Schätze gehoben
Zeile 1060-1280:	Subroutine Zusammenstoß mit Geist
Zeile 1290-1320:	Subroutine Ton initialisieren
Zeile 1330-1680:	Subroutine Spielanleitung
Zeile 1690-1730:	Subroutine Spielfeld aufbauen
Zeile 1740-3300:	Subroutine Leitern stellen
Zeile 3310-3440:	Subroutine Schätze verteilen
Zeile 3450-3790:	Maschinenprogramm Bildschirmaufbau
Zeile 3800-5970:	Maschinenprogramm Bewegung/ Tastaturabfrage
Zeile 5980-7950:	Spriteformen

Das Programm »Schatzsucher« nach Zeilennummern aufgeschlüsselt

2. Erläuterungen:
Da Video Bank 2 gewählt wurde, muß dies dem Screen Editor mit POKE 648,132 mitgeteilt werden.
Wird ein Programm nach Zeile 220 mit ⟨RUN/STOP⟩ ⟨RESTORE⟩ abgebrochen, ist auf dem Bildschirm nichts Sinnvolles zu erkennen, da der Screen-Editor nicht automatisch zurückgesetzt wird. Dies kann jedoch durch ein POKE 648,4 manuell geschehen.
Beim Einlesen der DATA-Zeilen des Maschinenprogrammes wird alle 10 Zeilen ein Checksummentest durchgeführt, um Eingabefehler zu vermeiden. Dieser Test erkennt jedoch keine Fehler, die sich gegenseitig aufheben.
Wenn das Programm einwandfrei läuft, kann dieser Test entfernt werden, um das Einlesen der Data-Zeilen zu verkürzen. Am vorteilhaftesten ist es, wenn die beiden Maschinenprogramme mit Hilfe eines Monitors als Maschinenprogramm abgespeichert werden, da dies die Ladezeit gewaltig verringert.

$C000-$C0C4:	Programm zum Bildschirmaufbau
$C400-$C92E:	Programm Bewegung/ Tastaturabfrage
Temporäre Speicher:	$CF90-$CFFF
Zeiger:	$FB, $FC, $FD, $FE

Einige Erläuterungen zum »Schatzsucher«

zsucher

Eine Spielszene vom »Schatzsucher«

pchen, zeigen
lem — bewahren Sie ruhig Blut.

Listing des Monats: Schatzsucher

Das Spiel Schatzsucher braucht sich nicht hinter professionellen Spielen zu verstecken. Wegen der grafisch ansprechenden Lösung und dem Spielwitz wurde es zum Listing des Monats gewählt.

Der Autor Gerhard Klauser stellt sich im folgenden kurz selbst vor:

— Geboren am 8.12.1958 in Dornbirn/Österreich
— Ausbildung als Elektrotechniker
— Derzeitige Tätigkeit: Testprogrammerstellung für Leiterplatten und Prüfplanung
— Zur Spielidee: Als ich im Sommer 1983 den Commodore 64 erwarb, da waren nur wenige Spielprogramme erhältlich. Daher beschloß ich, ein Spiel, das ich auf einem Spielautomaten gesehen hatte, in verbesserter Form für den C 64 zu erstellen. Das Ergebnis von zirka 60 Stunden Arbeit liegt Ihnen nun vor. Anmerkung: Das Spiel war ursprünglich vollkommen in Basic geschrieben, es war jedoch so langsam, so daß ich die Bewegungsabläufe in einem Maschinenprogramm zusammenfaßte.

(Ing. Gerhard Klauser)

```
10 REM SCHATZSUCHER
20 POKE56,127
25 DIM LH(6,2):DIMLE(6,2)
30 FORX=0TO6:FORY=0TO2
40 LH(X,Y)=0
50 LE(X,Y)=0
60 NEXTY
70 NEXTX
80 GOSUB1330:GOSUB650
90 PU=0:DU=0:AS=1:PG=0
100 BA=32768
110 FORX=0TO62:READDA:POKE15808+BA+X,DA:NEXT
120 FORX=0TO62:READDA:POKE15872+BA+X,DA:NEXT
130 FORX=0TO62:READDA:POKE15936+BA+X,DA:NEXT
140 FORX=0TO62:READDA:POKE16000+BA+X,DA:NEXT
150 FORX=0TO62:READDA:POKE16064+BA+X,DA:NEXT
160 FORX=0TO62:READDA:POKE16128+BA+X,DA:NEXT
170 FORX=0TO62:READDA:POKE832+BA+X,DA:NEXT
180 FORX=0TO62:READDA:POKE896+BA+X,DA:NEXT
190 FORX=0TO62:READDA:POKE960+BA+X,DA:NEXT
200 OT=PEEK(56578)
210 POKE56578,PEEK(56578)OR3:POKE56576,(PEEK(56576)AND252)OR1
220 POKE56578,OT:POKE648,132:PRINT"[CLR]"
230 WE=42+128:POKE53150,WE
240 POKE2040+BA,247:POKEV+21,3
250 V=53248:POKEV,32:POKEV+1,220:POKEV+21,1:POKEV+39,7:XM=32:YM=220:POKE2040+BA,
2
260 POKEV+16,0
270 POKEV+37,2:POKEV+38,7:POKEV+39,6:POKEV+40,5:POKEV+41,4:POKEV+42,3:POKEV+43,7
280 POKEV+28,255
290 PU=0:DU=0:AS=1:PG=0
300 POKEV+21,3:POKE2040+BA,247
310 GOSUB1690:POKE53149,0
320 POKEV+16,PEEK(V+16)AND254:POKEV,32:POKEV+1,220:YM=220:XM=32:POKE2040+BA,247
330 GOSUB1740
340 GOSUB3310
350 PO=BA+1024+INT((YM-50)/8)*40+INT((XM-24)/8)
360 PH=INT(PO/256):PL=PO-PH*256
370 POKE53238,PL:POKE53239,PH
380 GOSUB490
390 POKE53265,PEEK(53265)OR16
400 GOSUB1290
410 REM MANN BEWEGEN/ABFRAGE
420 SYS50176
430 FORX=1TOSG:NEXT
```

Listing: »Schatzsucher«

```
450 IF(PEEK(V+30)AND(1))=1THEN1060
460 PU=PU+PEEK(53149):POKE53149,0
470 IFPU=10THEN780
480 GOTO 420
490 REM GEIST INITIALISIEREN
500 X(1)=32:Y(1)=58:X(2)=312:Y(2)=58:X(3)=32:Y(3)=90:
    X(4)=312:Y(4)=90
510 POKEV+2,X(1):POKEV+3,Y(1)
520 POKEV+4,X(2)-255:POKEV+5,Y(2)
530 POKEV+6,X(3):POKEV+7,Y(3)
540 POKEV+8,X(4)-255:POKEV+9,Y(4):POKEV+16,20
550 POKE2041+BA,13:V(1)=0:POKE2042+BA,14:V(2)=1
560 POKE2043+BA,15:V(3)=2:POKE2044+BA,13:V(4)=0
570 FORK=1TO4:R(K)=2:NEXT
580 FORK=1TO4
590 PO(K)=BA+1024+INT((Y(K)-50)/8)*40+INT((X(K)-24)/8):NEXTK
600 FORK=1TO4:POKE53239+2*K,(INT(PO(K)/256))
610 POKE53238+2*K,(PO(K)-(INT(PO(K)/256))*256):NEXTK
620 POKE53229+2,37:POKE53229+4,34:POKE53229+6,37:POKE53229+8,34
630 POKE53219+2,37:POKE53219+4,34:POKE53219+6,37:POKE53219+8,34
640 POKEV+30,0:POKE54272+14,255:POKE54272+15,255:POKE54272+18,128:RETURN
650 REM LADEN DES MASCHINENPROGRAMMS
660 ZIEL=49152:XS=0
670 READX:IFX=-1THEN750
680 XS=XS+X
690 IF X<>(-2)THEN GOTO 740
700 READX$:READ CHECKSUM:IFXS=CHECKSUMTHENXS=0:GOTO670
710 PRINT"";"CHECKSUMMENFEHLER IN ";X$
720 PRINT"SOLLWERT=";CHECKSUM
730 PRINT"ISTWERT =";XS:STOP
740 POKEZIEL,X:ZIEL=ZIEL+1:GOTO670
750 IF ZIEL>51473THENRETURN
760 ZIEL=50176:GOTO670
770 RETURN
780 REM ALLE SCHAETZE GEHOBEN
790 PG=(PU*AS*100)*PU(0)+(PG)
800 PU=0:POKE53149,0
810 IF MA<PGTHENMA=PG
820 V2=PEEK(V+21)
830 GOSUB920
840 PRINT"IHRE PUNKTE:   ",PG
850 PRINT"MAXIMALE PUNKTE:",MA
860 PRINT"SIE HABEN NOCH ",(4-DU);" SCHATZSUCHER"
870 IF DU=4 THEN PRINT"SPIELENDE":FORK=1TO5000:NEXTK:GOTO200
880 FORK=1TO5000:NEXTK:AS=AS+1
890 IFAS=2THENPOKEV+21,7:GOTO310
900 IFAS=3THENPOKEV+21,15:GOTO310
910 POKEV+21,31:GOTO310
920 SI=54272:POKESI+12,9:POKESI+13,255:POKESI+24,15:POKESI+1[illegible]
930 FORX=0TO255STEP15:POKESI+8,X:POKESI+7,255
940 FORY=0TO255STEP15:POKESI+7,255-Y
950 NEXTY
960 NEXTX
970 POKESI+12,9:POKESI+13,0:POKESI+8,10:POKESI+7,177
980 IF V2=31 THEN UM=4
990 IF V2=15 THEN UM=3
1000 IF V2=07 THEN UM=2
1010 IF V2=03 THEN UM=1
1020 FORX=UM TO 0 STEP-1
1030 POKESI+11,33:POKESI+11,32
1040 POKEV+21,(PEEK(V+21)AND(255-2↑X))
1050 FORY=0TO800:NEXTY:NEXTX:POKEV+21,0:PRINT"":RETURN
1060 REM ZUSAMMENSTOSS MIT GEIST
1070 SI=54272:POKESI+12,9:POKESI+13,255:POKESI+24,15:POKESI+8,8:POKESI+7,255:POK
ESI+11,17
1080 V2=PEEK(V+21):POKEV+21,V2AND254
1090 FORZE=1TO10:POKEV+29,255:POKEV+23,255:POKESI+8,20
1100 FORWA=1TO70:NEXTWA:POKEV+29,0:POKEV+23,0:POKESI+8,8:FORWA=1TO70:NEXTWA:NEXT
ZE
1110 POKEV+29,0:POKEV+23,0:POKESI+11,0
1120 PG=(PU*AS*100)*PU(0)+PG
1130 PU=0:DU=DU+1
1140 IF MA<PGTHENMA=PG
1150 PRINT""
1160 POKEV+21,0
1170 PRINT"IHRE PUNKTE:   ",PG
1180 PRINT"MAXIMALE PUNKTE:",MA
1190 PRINT"SIE HABEN NOCH ",(4-DU);" SCHATZSUCHER"
1200 IFDU=4 THEN PRINT"SPIELENDE":FORK=1TO1000:NEXTK:GOTO1220
1210 GOTO1270
1220 POKE198,0:INPUT"SCHWIERIGKEITSGRAD ANGEBEN (1-10)";SG
1230 PU(0)=INT(SG)
1240 IFPU(0)>10THENPU(0)=10
1250 IFPU(0)<1THENPU(0)=1
1260 SG=100-10*PU(0):GOTO200
1270 FORK=1TO5000:NEXTK
1280 POKEV+21,V2:GOTO310
1290 REM TON
1300 SI=54272:FL=SI:FH=SI+1:TL=SI+2:TH=SI+3:W=SI+4:A=SI+5:H=SI+6:L=SI+24
1310 POKEL,15:POKEA,15*16+15:POKEH,1*16+1:POKETH,8:POKETL,0:POKEFH,14:POKEFL,162
1320 RETURN
1330 REM SPIELANLEITUNG
1340 POKE53280,2:POKE53281,2:PRINT""
1350 PRINT"COPYRIGHT COPYRIGHT COPYRIGHT COPYRIGHT"
```

Listing »Schatzsucher« (Fortsetzung)

Vorgezogener Schluß des Listing (von Seite 95) »Schatzsucher«

(von Seite 95)

```
6910 DATA 000,020,128
6920 DATA 000,020,128
6930 DATA 000,170,128
6940 DATA 002,170,000
6950 DATA 002,170,000
6960 DATA 008,170,000
6970 DATA 008,170,000
6980 DATA 004,170,000
6990 DATA 000,170,000
7000 DATA 000,255,000
7010 DATA 000,255,000
7020 DATA 000,195,000
7030 DATA 000,195,000
7040 DATA 000,131,000
7050 DATA 002,131,000
7060 DATA 000,002,000
7070 DATA 000,002,128
7080 REM MANN STEIGEND
     LINKS
7090 DATA 000,060,000
7100 DATA 004,255,000
7110 DATA 008,020,000
7120 DATA 008,020,000
7130 DATA 002,020,000
7140 DATA 002,020,000
7150 DATA 002,170,000
7160 DATA 000,170,128
7170 DATA 000,170,128
7180 DATA 000,170,032
7190 DATA 000,170,032
7200 DATA 000,170,016
7210 DATA 000,170,000
7220 DATA 000,255,000
7230 DATA 000,255,000
7240 DATA 000,195,128
7250 DATA 000,195,128
7260 DATA 000,194,000
7270 DATA 000,194,128
7280 DATA 000,128,000
7290 DATA 002,128,000
7300 REM GEIST FORM 1
7310 DATA 000,040,000
7320 DATA 000,040,000
7330 DATA 000,170,000
7340 DATA 002,170,128
7350 DATA 002,105,128
7360 DATA 138,170,162
7370 DATA 170,170,170
7380 DATA 169,105,106
7390 DATA 170,085,170
7400 DATA 138,085,162
7410 DATA 010,150,160
7420 DATA 002,170,128
7430 DATA 002,170,128
7440 DATA 002,170,128
7450 DATA 000,170,000
7460 DATA 000,130,000
7470 DATA 000,130,000
7480 DATA 002,162,000
7490 DATA 000,002,000
7500 DATA 000,002,000
7510 DATA 000,010,128
7520 REM GEIST FORM 2
7530 DATA 000,040,000
7540 DATA 032,040,008
7550 DATA 168,170,042
7560 DATA 170,170,170
7570 DATA 042,105,168
7580 DATA 010,105,160
7590 DATA 010,105,160
7600 DATA 010,170,160
7610 DATA 010,170,160
7620 DATA 010,085,160
7630 DATA 010,150,160
7640 DATA 010,085,160
7650 DATA 002,170,128
7660 DATA 002,170,128
7670 DATA 000,170,000
7680 DATA 000,132,000
7690 DATA 000,132,000
7700 DATA 000,132,000
7710 DATA 000,132,000
7720 DATA 000,132,000
7730 DATA 001,170,128
7740 REM GEIST FORM 3
7750 DATA 000,040,000
7760 DATA 000,040,000
7770 DATA 000,170,000
7780 DATA 002,170,128
7790 DATA 002,105,128
7800 DATA 010,105,160
7810 DATA 010,170,160
7820 DATA 041,150,104
7830 DATA 041,085,104
7840 DATA 169,085,108
7850 DATA 170,105,170
7860 DATA 138,170,162
7870 DATA 130,170,130
7880 DATA 002,170,128
7890 DATA 000,170,000
7900 DATA 000,130,000
7910 DATA 000,130,000
7920 DATA 000,138,128
7930 DATA 000,128,000
7940 DATA 000,128,000
7950 DATA 002,160,000
READY.
```

```
1360 PRINT"████"
1370 PRINT"█████████S C H A T Z S U C H E"
1380 PRINT"████████████████ING. KLAUSER G.
1390 PRINT"█████████████6850 DORNBIRN
1400 PRINT"█████████████HAEHLINGEN 8
1410 PRINT"████████████████AUSTRIA"
1420 PRINT"█████COPYRIGHT COPYRIGHT COPYRIGHT COPYRIGHT"
1430 FORK=1TO3000:NEXTK
1440 PRINT"█"
1450 PRINT"ES GEHT DARUM, MOEGLICHST VIELE SCHAETZE";
1460 PRINT"ZU HEBEN, OHNE VON DEN GEISTERN GEFANGEN";
1470 PRINT"ZU WERDEN."
1480 PRINT"WENN ALLE SCHAETZE GEHOBEN SIND, DANN"
1490 PRINT"ERHALTEN SIE EIN FREISPIEL."
1500 PRINT"ZUSAETZLICH ERHOEHT SICH DER SCHWIERIG- KEITSGRAD."
1510 PRINT"BEI KONTAKT MIT EINEM GEIST WIRD NEU"
1520 PRINT"GESTARTET, DIE ANZAHL DER SCHATZSUCHER"
1530 PRINT"WIRD UM EINS VERRINGERT"
1540 PRINT"BEIM START STEHEN 4 SCHATZSUCHER ZUR VERFUEGUNG."
1550 PRINT"█"
1560 PRINT"███████████B E W E G U N G :"
1570 PRINT"███████████================="
1580 PRINT"███████████K P >  =  AUFWAERTS"
1590 PRINT"███████████K. >  =  ABWAERTS"
1600 PRINT"███████████KL >  =  LINKS"
1610 PRINT"███████████K; >  =  RECHTS"
1620 PRINT"███████████ODER     JOYSTICK 2"
1630 INPUT"█SCHWIERIGKEITSGRAD ANGEBEN (1-10)";SG
1640 PU(0)=INT(SG)
1650 IFPU(0)>10THENPU(0)=10
1660 IFPU(0)<1THENPU(0)=1
1670 SG=100-10*PU(0)
1680 RETURN
1690 REM SPEILFELD AUFBAUEN
1700 PRINT"█"
1710 SYS49152
1720 POKE53280,0:POKE53281,0
1730 RETURN
1740 REM LEITERN AUFBAUEN
1750 LA(2)=INT(RND(4)*3)+1
1760 IFLA(2)=1THEN1860
1770 LE(2,2)=INT(RND(5)*31)+1308+BA
1780 FORX=0TO3
1790 POKELE(2,2)-X*40,115
1800 POKELE(2,2)-X*40+54272-BA,10
1810 POKELE(2,2)-1-X*40,64
1820 POKELE(2,2)-1-X*40+54272-BA,10
1830 POKELE(2,2)-2-X*40,107
1840 POKELE(2,2)-2-X*40+54272-BA,10
1850 NEXTX
1860 LE(2,1)=INT(RND(5)*31)+1308+BA
1870 IFLE(2,1)>(LE(2,2)+8)THEN1900
1880 IFLE(2,1)<(LE(2,2)-11)THEN1900
1890 GOTO1860
1900 FORX=0TO3
1910 POKELE(2,1)-X*40,115
1920 POKELE(2,1)-X*40+54272-BA,10
1930 POKELE(2,1)-1-X*40,64
1940 POKELE(2,1)-1-X*40+54272-BA,10
1950 POKELE(2,1)-2-X*40,107
1960 POKELE(2,1)-2-X*40+54272-BA,10
1970 NEXT
1980 LA(3)=2
1990 IFLA(3)=1THEN2190
2000 LH(3,2)=INT(RND(6)*2)+1
2010 LE(3,2)=INT(RND(2)*31)+1468+BA
2020 IFLE(3,2)>LE(2,1)+5+160THEN2050
2030 IFLE(3,2)<LE(2,1)-7+160THEN2050
2040 GOTO2010
2050 IFLE(3,2)>LE(2,2)+5+160THEN2080
2060 IFLE(3,2)<LE(2,2)-7+160THEN2080
2070 GOTO2010
2080 FORX=1TO4*LH(3,2)
2090 POKELE(3,2)-(X-1)*40,115
2100 POKELE(3,2)-(X-1)*40+54272-BA,10
2110 POKELE(3,2)-1-(X-1)*40,64
2120 POKELE(3,2)-1-(X-1)*40+54272-BA,10
2130 POKELE(3,2)-2-(X-1)*40,107
2140 POKELE(3,2)-2-(X-1)*40+54272-BA,10
2150 NEXT
2160 LH(3,1)=INT(RND(6)*2)+1
2170 LE(3,1)=INT(RND(2)*31)+1468+BA
2180 IFLE(3,1)>LE(2,1)+3+160THEN2210
2190 IFLE(3,1)<LE(2,1)-3+160THEN2210
2200 GOTO2170
2210 IFLE(3,1)>LE(2,2)+5+160THEN2240
2220 IFLE(3,1)<LE(2,2)-7+160THEN2240
2230 GOTO2170
2240 IFLE(3,1)>LE(3,2)+5THEN2270
2250 IFLE(3,1)<LE(3,2)-7THEN2270
2260 GOTO2170
2270 FORX=1TO4*LH(3,1)
2280 POKELE(3,1)-(X-1)*40,115
2290 POKELE(3,1)-(X-1)*40+54272-BA,10
2300 POKELE(3,1)-1-(X-1)*40,64
2310 POKELE(3,1)-1-(X-1)*40+54272-BA,10
2320 POKELE(3,1)-2-(X-1)*40,107
2330 POKELE(3,1)-2-(X-1)*40+54272-BA,10
2340 NEXT
2350 LA(4)=INT(RND(1)*2)+1
2360 IFLA(4)=1THEN2470
2370 LH(4,2)=1
2380 LE(4,2)=INT(RND(2)*31)+1628+BA
2390 FORX=1TO4*LH(4,2)
2400 POKELE(4,2)-(X-1)*40,115
2410 POKELE(4,2)-(X-1)*40+54272-BA,10
2420 POKELE(4,2)-1-(X-1)*40,64
2430 POKELE(4,2)-1-(X-1)*40+54272-BA,10
2440 POKELE(4,2)-2-(X-1)*40,107
2450 POKELE(4,2)-2-(X-1)*40+54272-BA,10
2460 NEXT
2470 LE(4,1)=INT(RND(2)*31)+1628+BA
2480 LH(4,1)=1
2490 IFLE(4,1)>LE(4,2)+7THEN2520
2500 IFLE(4,1)<LE(4,2)-10THEN2520
2510 GOTO2470
2520 FORX=1TO4*LH(4,1)
2530 POKELE(4,1)-(X-1)*40,115
2540 POKELE(4,1)-(X-1)*40+54272-BA,10
2550 POKELE(4,1)-1-(X-1)*40,64
2560 POKELE(4,1)-1-(X-1)*40+54272-BA,10
2570 POKELE(4,1)-2-(X-1)*40,107
2580 POKELE(4,1)-2-(X-1)*40+54272-BA,10
2590 NEXT
2600 LA(5)=INT(RND(7)*4)+1
2610 IFLA(5)=1THEN2790
2620 LH(5,2)=INT(RND(6)*4)+1
2630 IFLH(5,2)>2THENLH(5,2)=2
2640 LE(5,2)=INT(RND(2)*31)+1788+BA
2650 IFLE(5,2)>LE(4,1)+5+160THEN2680
2660 IFLE(5,2)<LE(4,1)-7+160THEN2680
2670 GOTO2640
2680 IFLE(5,2)>LE(4,2)+5+160THEN2710
2690 IFLE(5,2)<LE(4,2)-7+160THEN2710
2700 GOTO2640
2710 FORX=1TO4*LH(5,2)
2720 POKELE(5,2)-(X-1)*40,115
2730 POKELE(5,2)-(X-1)*40+54272-BA,10
2740 POKELE(5,2)-1-(X-1)*40,64
2750 POKELE(5,2)-1-(X-1)*40+54272-BA,10
2760 POKELE(5,2)-2-(X-1)*40,107
2770 POKELE(5,2)-2-(X-1)*40+54272-BA,10
2780 NEXT
2790 REM
2800 LH(5,1)=INT(RND(6)*2)+1
2810 LE(5,1)=INT(RND(2)*31)+1788+BA
2820 IFLE(5,1)>LE(4,1)+3+160THEN2850
2830 IFLE(5,1)<LE(4,1)-3+160THEN2850
2840 GOTO2810
2850 IFLE(5,1)>LE(4,2)+3+160THEN2880
2860 IFLE(5,1)<LE(4,2)-3+160THEN2880
2870 GOTO2810
2880 IFLE(5,1)>LE(5,2)+8THEN2910
2890 IFLE(5,1)<LE(5,2)-7THEN2910
2900 GOTO2810
2910 FORX=1TO4*LH(5,1)
2920 POKELE(5,1)-(X-1)*40,115
2930 POKELE(5,1)-(X-1)*40+54272-BA,10
2940 POKELE(5,1)-1-(X-1)*40,64
2950 POKELE(5,1)-1-(X-1)*40+54272-BA,10
2960 POKELE(5,1)-2-(X-1)*40,107
2970 POKELE(5,1)-2-(X-1)*40+54272-BA,10
2980 NEXT
2990 LH(6,2)=1
3000 LA(6)=2
3010 IFLA(6)=1THEN3170
3020 LE(6,2)=INT(RND(2)*31)+1948+BA
3030 IFLE(6,2)>LE(5,1)+5+160THEN3060
3040 IFLE(6,2)<LE(5,1)-7+160THEN3060
3050 GOTO3020
3060 IFLE(6,2)>LE(5,2)+5+160THEN3090
3070 IFLE(6,2)<LE(5,2)-7+160THEN3090
3080 GOTO3020
3090 FORX=1TO4*LH(6,2)
3100 POKELE(6,2)-(X-1)*40,115
3110 POKELE(6,2)-(X-1)*40+54272-BA,10
3120 POKELE(6,2)-1-(X-1)*40,64
3130 POKELE(6,2)-1-(X-1)*40+54272-BA,10
3140 POKELE(6,2)-2-(X-1)*40,107
3150 POKELE(6,2)-2-(X-1)*40+54272-BA,10
3160 NEXT
3170 LH(6,1)=1
3180 LE(6,1)=INT(RND(2)*31)+1948+BA
3190 IFLE(6,1)>LE(6,2)+8THEN3220
3200 IFLE(6,1)<LE(6,2)-7THEN3220
3210 GOTO3180
3220 FORX=1TO4*LH(6,1)
3230 POKELE(6,1)-(X-1)*40,115
```

Listing »Schatzsucher« (Fortsetzung)

```
3240 POKELE(6,1)-(X-1)*40+54272-BA,10
3250 POKELE(6,1)-1-(X-1)*40,64
3260 POKELE(6,1)-1-(X-1)*40+54272-BA,10
3270 POKELE(6,1)-2-(X-1)*40,107
3280 POKELE(6,1)-2-(X-1)*40+54272-BA,10
3290 NEXT
3300 RETURN
3310 REM SCHAETZE VERTEILEN
3320 AZ=0
3330 FORX=0TO9
3340 PS=INT(RND(8)*37)+2
3350 ZE=INT(RND(4)*6)+1
3360 IFPEEK(1104+BA+PS+(ZE-1)*160)=32THEN3380
3370 GOTO3340
3380 IFPEEK(1104+BA+PS-1+(ZE-1)*160)=160THEN3340
3390 IFPEEK(1104+BA+PS+1+(ZE-1)*160)=160THEN3340
3400 POKE(1104+BA+PS+(ZE-1)*160),WE
3410 POKE(1104+PS+(ZE-1)*160+54272),13
3420 AZ=AZ+1
3430 NEXTX
3440 RETURN
3450 REM MP DATA ZEILEN
3460 DATA 160, 0, 162, 0, 173, 194
3470 DATA 192, 133, 251, 173, 195, 192
3480 DATA 133, 252, 169, 13, 145, 251
3490 DATA 230, 251, 240, 3, 76, 27
3500 DATA 192, 230, 252, 165, 252, 201
3510 DATA 219, 240, 3, 76, 14, 192
3520 DATA 165, 251, 201, 255, 240, 3
3530 DATA 76, 14, 192, 234, 169, 0
3540 DATA 133, 251, 169, 132, 133, 252
3550 DATA 160, 0, 32, 181, 192, 169, -2
   , "3460 - 3550", 9253
3560 DATA 160, 133, 251, 169, 132, 133
3570 DATA 252, 160, 0, 32, 181, 192
3580 DATA 169, 64, 133, 251, 169, 133
3590 DATA 133, 252, 160, 0, 32, 181
3600 DATA 192, 169, 224, 133, 251, 169
3610 DATA 133, 133, 252, 160, 0, 32
3620 DATA 181, 192, 169, 128, 133, 251
3630 DATA 169, 134, 133, 252, 160, 0
3640 DATA 32, 181, 192, 169, 32, 133
3650 DATA 251, 169, 135, 133, 252, 160, -2
   , "3560 - 3650", 9059
3660 DATA 0, 32, 181, 192, 169, 192
3670 DATA 133, 251, 169, 135, 133, 252
3680 DATA 160, 0, 32, 181, 192, 162
3690 DATA 0, 169, 0, 133, 251, 169
3700 DATA 132, 133, 252, 160, 39, 169
3710 DATA 160, 145, 251, 200, 145, 251
3720 DATA 232, 224, 24, 240, 19, 169
3730 DATA 40, 24, 101, 251, 176, 5
3740 DATA 133, 251, 76, 147, 192, 133
3750 DATA 251, 230, 252, 76, 147, 192
3760 DATA 96, 169, 160, 145, 251, 200
3770 DATA 192, 40, 240, 3, 76, 181
3780 DATA 192, 96, 00, 216
3790 DATA -2, "3660 - 3790", 11165, -1
3800 DATA 173, 159, 207, 201, 170, 240
3810 DATA 3, 32, 19, 200, 169, 255
3820 DATA 141, 2, 220, 165, 203, 201
3830 DATA 41, 208, 5, 169, 33, 76
3840 DATA 56, 196, 201, 44, 208, 5
3850 DATA 169, 36, 76, 56, 196, 201
3860 DATA 42, 208, 5, 169, 34, 76
3870 DATA 56, 196, 201, 50, 208, 5
3880 DATA 169, 37, 76, 56, 196, 32
3890 DATA 158, 200, 162, 254, 142, 192, -2
   , "3800 - 3890" , 7658
3900 DATA 207, 162, 1, 160, 0, 140
3910 DATA 160, 207, 142, 176, 207, 153
3920 DATA 208, 207, 32, 1, 199, 144
3930 DATA 52, 32, 236, 196, 169, 17
3940 DATA 141, 4, 212, 169, 16, 141
3950 DATA 4, 212, 32, 88, 200, 160
3960 DATA 41, 173, 246, 207, 133, 253
3970 DATA 173, 247, 207, 133, 254, 177
3980 DATA 253, 205, 158, 207, 240, 8
3990 DATA 169, 0, 141, 157, 207, 76, -2
   , "3900 - 3990", 8650
4000 DATA 131, 196, 169, 32, 145, 253
4010 DATA 169, 1, 141, 157, 207, 160
4020 DATA 2, 140, 160, 207, 162, 253
4030 DATA 142, 192, 207, 162, 2, 142
4040 DATA 176, 207, 32, 22, 198, 32
4050 DATA 236, 196, 160, 1, 32, 72
4060 DATA 200, 160, 4, 140, 160, 207
4070 DATA 162, 251, 142, 192, 207, 162
4080 DATA 4, 142, 176, 207, 32, 22
4090 DATA 198, 32, 236, 196, 160, 2, -2
   , "4000 - 4090", 8288
4100 DATA 32, 72, 200, 160, 6, 140
4110 DATA 160, 207, 162, 247, 142, 192
4120 DATA 207, 162, 8, 142, 176, 207
4130 DATA 32, 22, 198, 32, 236, 196
4140 DATA 160, 3, 32, 72, 200, 160
4150 DATA 8, 140, 160, 207, 162, 239
4160 DATA 142, 192, 207, 162, 16, 142
4170 DATA 176, 207, 32, 22, 198, 32
4180 DATA 236, 196, 160, 4, 32, 72
4190 DATA 200, 96, 185, 208, 207, 153, -2
   , "4100 - 4190", 8156
4200 DATA 224, 207, 201, 33, 240, 14
4210 DATA 201, 36, 240, 13, 201, 37
4220 DATA 240, 12, 201, 34, 240, 11
4230 DATA 24, 96, 76, 16, 197, 76
4240 DATA 64, 197, 76, 112, 197, 76
4250 DATA 160, 197, 185, 246, 207, 56
4260 DATA 233, 40, 144, 10, 133, 253
4270 DATA 185, 247, 207, 133, 254, 76
4280 DATA 44, 197, 133, 253, 185, 247
4290 DATA 207, 56, 233, 1, 133, 254, -2, "4200 - 4290", 8499
4300 DATA 165, 253, 153, 246, 207, 165
4310 DATA 254, 153, 247, 207, 185, 208
4320 DATA 207, 153, 224, 207, 32, 208
4330 DATA 197, 96, 185, 246, 207, 24
4340 DATA 105, 40, 176, 10, 133, 253
4350 DATA 185, 247, 207, 133, 254, 76
4360 DATA 92, 197, 133, 253, 185, 247
4370 DATA 207, 24, 105, 1, 133, 254
4380 DATA 165, 253, 153, 246, 207, 165
4390 DATA 254, 153, 247, 207, 185, 208, -2
   , "4300 - 4390", 10520
4400 DATA 207, 153, 224, 207, 32, 218
4410 DATA 197, 96, 185, 246, 207, 24
4420 DATA 105, 1, 176, 10, 133, 253
4430 DATA 185, 247, 207, 133, 254, 76
4440 DATA 140, 197, 133, 253, 185, 247
4450 DATA 207, 24, 105, 1, 133, 254
4460 DATA 165, 253, 153, 246, 207, 165
4470 DATA 254, 153, 247, 207, 185, 208
4480 DATA 207, 153, 224, 207, 32, 228
4490 DATA 197, 96, 185, 246, 207, 56,-2, "4400-4490", 10134
4500 DATA 233, 1, 144, 10, 133, 253
4510 DATA 185, 247, 207, 133, 254, 76
4520 DATA 188, 197, 133, 253, 185, 247
4530 DATA 207, 56, 233, 1, 133, 254
4540 DATA 165, 253, 153, 246, 207, 165
4550 DATA 254, 153, 247, 207, 185, 208
4560 DATA 207, 153, 224, 207, 32, 253
4570 DATA 197, 96, 185, 1, 208, 56
4580 DATA 233, 8, 153, 1, 208, 96
4590 DATA 185, 1, 208, 24, 105, 8, -2, "4500 - 4590", 9453
4600 DATA 153, 1, 208, 96, 185, 0
4610 DATA 208, 24, 105, 8, 176, 4
4620 DATA 153, 0, 208, 96, 153, 0
4630 DATA 208, 173, 16, 208, 13, 176
4640 DATA 207, 141, 16, 208, 96, 185
4650 DATA 0, 208, 56, 233, 8, 144
4660 DATA 4, 153, 0, 208, 96, 153
4670 DATA 0, 208, 173, 16, 208, 45
4680 DATA 192, 207, 141, 16, 208, 96
4690 DATA 185, 224, 207, 201, 33, 208,-2, "4600-4690", 7254
4700 DATA 3, 76, 189, 198, 201, 36
4710 DATA 208, 3, 76, 189, 198, 192
4720 DATA 2, 208, 3, 76, 149, 198
4730 DATA 192, 6, 208, 3, 76, 149
4740 DATA 198, 173, 27, 212, 56, 233
4750 DATA 128, 144, 5, 169, 33, 76
4760 DATA 68, 198, 169, 36, 153, 208
4770 DATA 207, 32, 1, 199, 144, 3
4780 DATA 76, 255, 198, 185, 224, 207
4790 DATA 153, 208, 207, 32, 1, 199, -2, "4700 - 4790", 7654
4800 DATA 144, 3, 76, 255, 198, 201
4810 DATA 37, 208, 11, 169, 34, 153
4820 DATA 208, 207, 32, 1, 199, 76
4830 DATA 255, 198, 201, 34, 208, 11
4840 DATA 169, 37, 153, 208, 207, 32
4850 DATA 1, 199, 76, 255, 198, 201
4860 DATA 33, 208, 11, 169, 36, 153
4870 DATA 208, 207, 32, 1, 199, 76
4880 DATA 255, 198, 169, 33, 153, 208
4890 DATA 207, 32, 1, 199, 76, 255, -2, "4800 - 4890", 7972
4900 DATA 198, 173, 247, 207, 56, 249
4910 DATA 247, 207, 144, 17, 240, 3
4920 DATA 76, 182, 198, 173, 246, 207
4930 DATA 56, 249, 246, 207, 144, 3
4940 DATA 76, 182, 198, 234, 234, 169
4950 DATA 33, 76, 68, 198, 234, 234
4960 DATA 169, 36, 76, 68, 198, 185
4970 DATA 224, 207, 153, 208, 207, 32
4980 DATA 1, 199, 144, 3, 76, 255
4990 DATA 198, 173, 27, 212, 56, 233, -2, "4900-4990", 9249
5000 DATA 128, 144, 8, 169, 37, 153
5010 DATA 208, 207, 76, 224, 198, 169
5020 DATA 34, 153, 208, 207, 32, 1
```

Listing »Schatzsucher« (Fortsetzung)

```
5030 DATA 199, 144, 3, 76, 255, 198
5040 DATA 201, 37, 208, 11, 169, 34
5050 DATA 153, 208, 207, 32, 1, 199
5060 DATA 76, 255, 198, 169, 37, 153
5070 DATA 208, 207, 32, 1, 199, 56
5080 DATA 96, 185, 208, 207, 201, 33
5090 DATA 240, 14, 201, 36, 240, 13, -2, "5000 - 5090", 7954
5100 DATA 201, 37, 240, 12, 201, 34
5110 DATA 240, 11, 24, 96, 76, 153
5120 DATA 199, 76, 206, 199, 76, 92
5130 DATA 199, 185, 246, 207, 56, 233
5140 DATA 1, 144, 10, 133, 251, 185
5150 DATA 247, 207, 133, 252, 76, 59
5160 DATA 199, 133, 251, 185, 247, 207
5170 DATA 56, 233, 1, 133, 252, 140
5180 DATA 18, 201, 160, 0, 177, 251
5190 DATA 201, 160, 240, 19, 160, 40, -2, "5100 - 5190", 8659
5200 DATA 177, 251, 201, 160, 240, 11
5210 DATA 160, 80, 177, 251, 201, 160
5220 DATA 240, 3, 76, 3, 200, 76
5230 DATA 11, 200, 185, 246, 207, 24
5240 DATA 105, 1, 176, 10, 133, 251
5250 DATA 185, 247, 207, 133, 252, 76
5260 DATA 120, 199, 133, 251, 185, 247
5270 DATA 207, 24, 105, 1, 133, 252
5280 DATA 140, 18, 201, 160, 2, 177
5290 DATA 251, 201, 160, 240, 19, 160, -2, "5200 - 5290", 8900
5300 DATA 42, 177, 251, 201, 160, 240
5310 DATA 11, 160, 82, 177, 251, 201
5320 DATA 160, 240, 3, 76, 3, 200
5330 DATA 76, 11, 200, 185, 246, 207
5340 DATA 56, 233, 40, 144, 10, 133
5350 DATA 251, 185, 247, 207, 133, 252
5360 DATA 76, 181, 199, 133, 251, 185
5370 DATA 247, 207, 56, 233, 1, 133
5380 DATA 252, 140, 18, 201, 160, 0
5390 DATA 177, 251, 201, 160, 240, 11, -2, "5300 - 5390", 9162
5400 DATA 200, 192, 3, 240, 3, 76
5410 DATA 186, 199, 76, 3, 200, 76
5420 DATA 11, 200, 185, 246, 207, 24
5430 DATA 105, 40, 176, 10, 133, 251
5440 DATA 185, 247, 207, 133, 252, 76
5450 DATA 234, 199, 133, 251, 185, 247
5460 DATA 207, 24, 105, 1, 133, 252
5470 DATA 140, 18, 201, 160, 80, 177
5480 DATA 251, 201, 160, 240, 11, 200
5490 DATA 192, 83, 240, 3, 76, 239, -2, "5400 - 5490", 8783
5500 DATA 199, 76, 3, 200, 76, 11
5510 DATA 200, 172, 18, 201, 185, 208
5520 DATA 207, 56, 96, 172, 18, 201
5530 DATA 185, 208, 207, 24, 96, 169
5540 DATA 170, 141, 159, 207, 160, 2
5550 DATA 169, 37, 153, 224, 207, 160
5560 DATA 6, 153, 224, 207, 169, 34
5570 DATA 160, 4, 153, 224, 207, 160
5580 DATA 8, 153, 224, 207, 169, 255
5590 DATA 141, 14, 212, 141, 15, 212, -2, "5500 - 5590", 8427
5600 DATA 169, 128, 141, 18, 212, 169
5610 DATA 15, 141, 24, 212, 169, 0
5620 DATA 141, 157, 207, 96, 185, 248
5630 DATA 135, 201, 15, 208, 2, 169
5640 DATA 12, 24, 105, 1, 153, 248
5650 DATA 135, 96, 185, 224, 207, 133
5660 DATA 253, 201, 34, 240, 42, 201
5670 DATA 37, 240, 19, 173, 248, 135
5680 DATA 201, 251, 240, 6, 169, 251
5690 DATA 141, 248, 135, 96, 169, 252, -2, "5600 - 5690", 8665
5700 DATA 141, 248, 135, 96, 173, 248
5710 DATA 135, 201, 247, 240, 6, 169
5720 DATA 247, 141, 248, 135, 96, 169
5730 DATA 248, 141, 248, 135, 96, 173
5740 DATA 248, 135, 201, 249, 240, 6
5750 DATA 169, 249, 141, 248, 135, 96
5760 DATA 169, 250, 141, 248, 135, 96
5770 DATA 169, 224, 141, 2, 220, 173
5780 DATA 0, 220, 41, 1, 208, 5
5790 DATA 169, 33, 76, 0, 201, 173, -2, "5700 - 5790", 9326
5800 DATA 0, 220, 41, 2, 208, 5
5810 DATA 169, 36, 76, 0, 201, 173
5820 DATA 0, 220, 41, 4, 208, 5
5830 DATA 169, 34, 76, 0, 201, 173
5840 DATA 0, 220, 41, 8, 208, 5
5850 DATA 169, 37, 76, 0, 201, 173
5860 DATA 1, 220, 41, 1, 208, 5
5870 DATA 169, 33, 76, 0, 201, 173
5880 DATA 1, 220, 41, 2, 208, 5
5890 DATA 169, 36, 76, 0, 201, 173, -2, "5800 - 5890", 5658
5900 DATA 1, 220, 41, 4, 208, 5
5910 DATA 169, 34, 76, 0, 201, 173
5920 DATA 1, 220, 41, 8, 208, 14
5930 DATA 169, 37, 141, 18, 201, 169
5940 DATA 255, 141, 2, 220, 173, 18
5950 DATA 201, 96, 169, 255, 141, 2
5960 DATA 220, 96,
5970 DATA -2, "5900 - 5970", 4346, -1
5980 REM MANN STEHEND RECHTS
5990 DATA 000,060,000
6000 DATA 000,255,000
6010 DATA 000,020,000
6020 DATA 000,021,000
6030 DATA 000,020,000
6040 DATA 000,020,000
6050 DATA 000,170,000
6060 DATA 002,170,128
6070 DATA 010,040,160
6080 DATA 008,040,036
6090 DATA 004,040,000
6100 DATA 000,040,000
6110 DATA 000,060,000
6120 DATA 000,060,000
6130 DATA 000,012,000
6140 DATA 000,012,000
6150 DATA 000,012,000
6160 DATA 000,012,000
6170 DATA 000,012,000
6180 DATA 000,008,000
6190 DATA 000,010,000
6200 REM MANN GEHEND RECHTS
6210 DATA 000,060,000
6220 DATA 000,255,000
6230 DATA 000,020,000
6240 DATA 000,021,000
6250 DATA 000,020,000
6260 DATA 000,020,000
6270 DATA 000,042,000
6280 DATA 000,042,160
6290 DATA 000,040,164
6300 DATA 000,040,000
6310 DATA 000,040,000
6320 DATA 000,040,000
6330 DATA 000,040,000
6340 DATA 000,060,000
6350 DATA 000,063,000
6360 DATA 000,051,192
6370 DATA 000,048,192
6380 DATA 000,048,192
6390 DATA 000,048,128
6400 DATA 000,032,160
6410 DATA 000,040,000
6420 REM MANN STEHEND LINKS
6430 DATA 000,060,000
6440 DATA 000,255,000
6450 DATA 000,020,000
6460 DATA 000,084,000
6470 DATA 000,020,000
6480 DATA 000,020,000
6490 DATA 000,170,000
6500 DATA 002,170,128
6510 DATA 010,040,160
6520 DATA 024,040,032
6530 DATA 000,040,016
6540 DATA 000,040,000
6550 DATA 000,040,000
6560 DATA 000,060,000
6570 DATA 000,060,000
6580 DATA 000,048,000
6590 DATA 000,048,000
6600 DATA 000,048,000
6610 DATA 000,048,000
6620 DATA 000,032,000
6630 DATA 000,160,000
6640 REM MANN GEHEND LINKS
6650 DATA 000,060,000
6660 DATA 000,255,000
6670 DATA 000,020,000
6680 DATA 000,084,000
6690 DATA 000,020,000
6700 DATA 000,020,000
6710 DATA 000,168,000
6720 DATA 010,168,000
6730 DATA 026,040,000
6740 DATA 000,040,000
6750 DATA 000,040,000
6760 DATA 000,040,000
6770 DATA 000,040,000
6780 DATA 000,060,000
6790 DATA 000,252,000
6800 DATA 003,216,000
6810 DATA 003,012,000
6820 DATA 003,012,000
6830 DATA 002,012,000
6840 DATA 010,008,000
6850 DATA 000,040,000
6860 REM MANN STEIGEND RECHTS
6870 DATA 000,060,000
6880 DATA 000,255,016
6890 DATA 000,020,032
6900 DATA 000,020,032
```


Listing »Schatzsucher« (Fortsetzung)

Das Listing wird auf Seite 92 vorgesetzt

BÜCHER

Das große Floppy-Buch

Bei dem großen Floppy-Buch handelt es sich um genau 326 Seiten geballte Information über die VC 1541-Floppy von Commodore. Diese Diskettenstation wurde speziell für die Homecomputer im Sinne von Problembeschreibung, VC 20 und Commodore 64 entwickelt. Der schon etwas fortgeschrittene Anwender, der seine VC 1541-Einheit intensiver als nur zum Abspeichern von Programmen nutzen will, erfährt durch das Originalhandbuch leider nur recht wenig Unterstützung. Das große Floppy-Buch deckt hier also einen echten Bedarf.

Neben der Beschreibung aller Floppy-Systembefehle zum Filehandling werden in zwei ausführlichen Kapiteln die Konzepte der sequentiellen und der relativen Datenspeicherung erläutert. Ein Abschnitt über »Programmierung für Fortgeschrittene« zeigt Möglichkeiten zum direkten Zugriff auf einzelne Sektoren einer Diskette und auf das Betriebssystem (DOS) der Diskettenstation. Natürlich wird auch der technische Aufbau und die interne Verwaltung einer Floppy-Station beschrieben.

Neben einer Unzahl von kleineren Beispielprogrammen findet man unter anderem auch eine komplette Adreßverwaltung, ein Disketten-Kopierprogramm und einen Disk-Monitor. Ein vollständiges, kommentiertes Disassemblerlisting des DOS 2.6 und eine Vielzahl nützlicher Tips und Tricks für das Arbeiten mit diesem DOS runden das große Floppy-Buch ab, das infolge der durchweg leicht verständlichen Darstellungsweise nicht nur für den Profi interessant erscheint. (ev)

Englisch/Szczepanowski: Das große Floppy-Buch, 326 Seiten, Data Becker GmbH, Düsseldorf 1983, 49 Mark.

Commodore 64 Der Computer für Einsteiger und Aufsteiger

Das Buch wendet sich an den absoluten Computerneuling, dem Begriffe wie »Datenverarbeitung« und »Programmierung« wenig oder gar nichts sagen und der sich mit seinem neuen Commodore 64 auch nur auf der Ebene der Anwenderprogrammierung (also ohne weitere Kenntnisse über Bits, Bytes oder Speicheradressen) beschäftigen will.

Wer also ein tiefergehendes Verständnis des C 64 sucht oder sich Tips und Tricks für die Programmierung (oder gar Tabellen spezieller POKEs und PEEKs) erhofft, dem kann dieses Buch nicht weiterhelfen.

Dem Neuling jedoch wird das Programmieren von den Grundlagen her beigebracht, und zwar Programmierung richtig verstanden im Sinne von Problembeschreibung, Analyse, Finden eines Algorithmus und schließlich Codierung desselben in einer Programmiersprache, zum Beispiel in Basic. Nach Einübung dieser Vorgehensweise anhand einiger kleiner Beispiele wird schließlich die Spracherweiterung Simons Basic für den C 64 vorgestellt. Man erfährt hier einiges über die im Simons Basic enthaltenen Sprachelemente zur strukturierten Programmierung. Der Abschnitt über Unterprogrammtechnik im Simons Basic ist dabei ermüdend lang geraten. Über volle 6 Seiten wird an trivialen Beispielen im wesentlichen vorgeführt, daß ein Unterprogramm immer wieder ins Hauptprogramm zurückkehrt. Aber dazu braucht man wirklich kein Simons Basic, das funktioniert genauso mit GOSUB.

Insgesamt gesehen bleibt der Teil des Buches, der sich mit dem Simons Basic und strukturierter Programmierung befaßt, leider nur Stückwerk. Dieser Abschnitt kann höchstens einen ersten, flüchtigen Einblick darstellen. Unbefriedigend ist natürlich auch, daß der Anwender, der nicht über Simons Basic verfügt, hier völlig allein gelassen wird. Immerhin gäbe es ja auch über strukturierte Programmierung im Standard-Basic einiges zu sagen. Der erste Teil des Buches ist demgegenüber als Programmierlehrgang für Anfänger recht gut gelungen — aber wirklich nur für Anfänger. (ev)

Hans Riedel / Franz Quinke: Commodore 64, 160 Seiten, Friedrich Kiehl Verlag 1983, ISBN 3-470-80421-6, 49,80 Mark.

VC 20-Anwenderhandbuch

Endlich ein Buch über den VC 20, das gleichermaßen den Anfänger wie den fortgeschrittenen Anwender anspricht. Neben einer ausführlichen und leicht verständlichen Einführung in die Basic-Programmierung findet der Anfänger hier auch eine Fülle von Tips und Tricks, um die Hard- und Softwareeigenschaften des VC 20 optimal für eigene Programme zu nutzen. Ein ganzes Kapitel beschäftigt sich mit den Möglichkeiten zur Steuerung von Spielen mittels Joystick, Paddle oder Tastatur. Ein anderer Abschnitt beschreibt Entwurf und Realisierung benutzerfreundlicher Inputroutinen.

Ist man mit den Grundlagen seines VC 20 einigermaßen vertraut, dann bieten die Kapitel über die Erzeugung von Grafiken und Tönen einen tieferen Einblick in das Innenleben des Volkscomputers. Erklärt und anhand von Beispielprogrammen demonstriert werden die Erzeugung selbstdefinierter Zeichen und — als Anwendung hiervon — die Programmierung der hochauflösenden Grafik und des Vielfarben-Modus. Sehr ausführlich dargestellt ist auch die Realisierung spezieller Toneffekte wie Schwebungen, Vibrato oder Tremolo. Bemerkenswert ist die recht umfangreiche Behandlung der wichtigsten Peripheriegeräte. Der Bogen spannt sich hier vom Farbbandwechsel beim VC 1515-Grafikdrucker bis zum Direktzugriff auf einzelne Sektoren bei der 1540-Floppy.

Zahlreiche Tabellen, Diagramme und Fotos tragen einiges zum guten Verständnis

des dargebotenen Stoffes bei. Ein umfangreicher Anhang enthält die Speicherbelegung des VC 20, die Funktionen des 6560-Video-Chips, alle Basic-Statements und Funktionen (auch für Diskettenbetrieb) und andere Tabellen.

Nach näheren Informationen über die verschiedenen Speicher- und Spracherweiterungsmodule sucht man im VC 20-Anwenderhandbuch jedoch vergebens. Ebenfalls nicht behandelt wird der Zugriff auf die Maschinenebene

Einige sprachliche Ungereimtheiten sind vermutlich auf die Übersetzung aus dem Amerikanischen zurückzuführen. So ist zum Beispiel die Rede von »weichen Sektoren« auf einer Diskette (gemeint sind durch Software erzeugte Sektoren) und zum SYS-Befehl findet man als Erklärung: »übergibt an ein Subsystem« (was immer das sein mag).

Alles in allem handelt es sich bei dem Buch jedoch um eine recht gute Einführung für den Anfänger, aber auch der fortgeschrittene Benutzer wird hier wahrscheinlich noch einiges an neuem Wissen finden. In jedem Fall ist das Buch als umfangreiches Nachschlagewerk zum VC 20-System zu empfehlen. (ev)

John Heilborn / Ran Talbott: VC 20 Anwenderhandbuch, 388 Seiten, McGraw-Hill Book Company, Hamburg 1983, ISBN 3-89028-004-8, 32,- Mark.

Commodore 64-Programmieren leicht gemacht

Dieses Buch ist gedacht für den Neuling, der sich einen Commodore 64 gekauft hat oder kaufen möchte. Für erfahrene Aufsteiger, die von einem anderen Gerät kommen, sollte es ebenfalls von Nutzen sein. Das von den Autoren gesteckte Ziel ist klar; es geht darum, in verständlicher Form die Haupteigenschaften des Commodore 64 und seiner Programmiersprache Basic zu beschreiben. Das Buch umfaßt eine große Zahl Programmbeispiele, von Testroutinen für neue Befehle bis hin zu längeren Programmen für Spiele, interessante Grafikdisplays, Umgang mit Daten auf Tonbandkassetten, Musik und Spritesteuerung. Die meisten Kapitel enthalten eine oder mehrere Aufgaben, die Ihre Fortschritte prüfen und zusätzliche Vorschläge liefern. Am Ende des jeweiligen Kapitels stehen die Lösungen der Aufgaben und können damit als Zusatzinformation genutzt werden. Eine wichtige (und ziemlich ungewöhnliche) Eigenschaft dieses Buches sind mehrere Kapitel über Debugging (Fehlersuche): Wie man dahinterkommt, warum ein Programm nicht läuft, und den Fehler behebt. Jeder, der selber Programme schreibt, wird diese Kapitel als besonders nützlich empfinden. Es werden auch ein paar Empfehlungen für das Verfassen gut strukturierter Programme — vor allem für die sinnvolle Nutzung von Unterprogrammen — gegeben, um das Programm in überschaubare Blöcke aufzuteilen. Ein Beispiel dazu habe ich in dem Artikel »Strukturiertes Programmieren« in dieser Ausgabe verwendet. Dieses Buch befaßt sich nicht mit allen Möglichkeiten des Commodore 64. Zum Beispiel wird der Umgang und die Arbeit mit dem Diskettenlaufwerk nicht behandelt, wohl aber, wie man Daten auf Kassette unterbringt. Ich sehe dieses Buch aber für den Anfänger als nützlich an. Es hilft dem Computer- und Commodore-Neuling sicher und oft auch in einer humorvollen Weise auf die Beine. Ich kann es deshalb empfehlen. (gk)

Ian Stewart, Robin Jones: Commodore 64 — Programmieren leicht gemacht, 264 Seiten, Birkhäuser 1984, ISBN 3-7643-1588-1, Preis 49 Mark.

Besser programmieren mit dem VC 20

Das »Buch zum Handbuch«, wie es im Untertitel genannt wird, wendet sich in erster Linie an den Computerneuling und will diesem über das Handbuch hinausgehende Hilfestellung bei der Programmierung seines Volkscomputers geben.

Thematisch gibt es drei Schwerpunkte: Das VC 20-Basic, die hochauflösende Grafik und Anregungen für die Erstellung eigener Software, hauptsächlich aus den Bereichen Grafik, Spiele und (naturwissenschaftliche) Simulationen. Die Vorschläge gehen jedoch kaum über das Niveau reiner Programmierübungen hinaus (Quiz, Nimm, Fibonacci-Zahlen, Tapetenmuster und so weiter).

Recht informativ ist der Abschnitt über die Realisierung von hochauflösenden Grafiken mit oder ohne Super-Expander, interessant auch die mit kleinen Beispielprogrammen beschriebenen Tricks, um Grafiken auf Band oder Disk zu retten oder als Hardcopy auf einem VC 1525-Drucker auszugeben.

Leider wirkt die Darstellung streckenweise etwas konfus und unübersichtlich. So erfährt der staunende Leser zum Beispiel, daß die Wertzuweisung mittels Gleichheitszeichen »zu den größten Handicaps von Basic« gehört oder daß der Computer als Ergebnis eines logischen Vergleichs A=B den Wert —1 liefert, »wenn über A und B noch nichts bekannt ist«. Speziell bei den Abschnitten über die hochauflösende Grafik wäre auch eine ausführlichere und systematischere Darstellung angebracht. Die Ausführungen über das VC 20-Basic gehen kaum über die des Handbuchs hinaus.

Fazit: Für den Profi bietet das Buch zuwenig, für den Neuling erscheint es jedoch trotz einiger Mängel in der Darstellung dann geeignet, wenn er sich für den Einsatz des VC 20 im Bereich Schule/Ausbildung interessiert. (ev)

Dr. Norbert Treitz: Besser programmieren mit dem VC 20, 186 Seiten, Lehrmittelverlag W. Hagemann 1983, ISBN 3-544-53001-0, 18,30 Mark.

64'er FUNDGRUBE

Wollen Sie einen gebrauchten Computer verkaufen oder erwerben? Suchen Sie Zubehör? Haben Sie Software anzubieten oder suchen Sie Programme oder Verbindungen? Die FUNDGRUBE von 64'er bietet allen Computerfans die Gelegenheit, für nur DM 5,— eine private Kleinanzeige mit bis zu 5 Zeilen Text in der Rubrik Ihrer Wahl aufzugeben. Und so kommt Ihre private Kleinanzeige in die FUNDGRUBE der **Juni-Ausgabe** (erscheint am 18. Mai 84): Schicken Sie Ihren Anzeigentext bis zum 24. April 84 (Datum des Poststempels und Anzeigenschluß) an »64'er«. Später eingehende Aufträge werden in der **Juli-Ausgabe** (erscheint am 22. Juni 84) veröffentlicht.

Am besten verwenden Sie dazu die vorbereitete Auftragskarte am Anfang des Heftes. Bitte beachten Sie: Ihr Anzeigentext darf maximal 5 Zeilen mit je 32 Buchstaben betragen. Überweisen Sie den Anzeigenpreis von DM 5,— auf das Postscheckkonto Nr. 14199-803 beim Postscheckamt mit dem Vermerk »Markt & Technik, 64'er«, oder schicken Sie uns DM 5,— als Scheck, in Briefmarken oder in Bargeld. Der Verlag behält sich die Veröffentlichung längerer Texte vor. Kleinanzeigen, die entsprechend gekennzeichnet sind oder deren Text auf eine gewerbliche Tätigkeit schließen läßt, werden in der Rubrik »Gewerbliche Kleinanzeigen« zum Preis von DM 10,— je Zeile Text veröffentlicht.

64'er

FUNDGRUBE

Teil 2

Alle Tasten-, Zeichen- und ASCII-Codes

Hier lesen Sie, wie man eine vollständige Tabelle der ASCII-Codes vom VC 20 und Commodore 64 erstellt. Außerdem werden Sie erfahren, daß die beiden Systeme wesentlich mehr Funktionstasten bieten als Sie bisher vielleicht angenommen haben.

Im ersten Teil habe ich Ihnen gezeigt, wie alle Tasten des VC 20 beziehungsweise des C 64 in einer Matrix angeordnet sind. Sobald eine Taste gedrückt wird, steht eine spezielle, nur dieser einen Taste zugeteilte Code-Zahl im Register 37152 des VC 20. Das entsprechende Register des C 64 ist 56320.

Ich habe auch erklärt, wie die Code-Zahlen zustande kommen. In einem kleinen Demonstrationsprogramm haben wir dann durch Abfragen dieses Registers mit PEEK bestimmte Programmschritte mit Tastendruck gesteuert.

Ein kurzer Rückblick

Wir haben auch herausgefunden, daß das Betriebssystem des Computers bei Verwendung von Basic eine Abfrage von nur acht Tasten zuläßt. Die Abfrage aller Tasten, auch mehrerer Tasten gleichzeitig, mit einem Programm in Maschinencode habe ich Ihnen für den Schluß versprochen.

Wir sind aber noch einen Schritt weitergegangen und haben herausgefunden, daß diese Code-Zahlen der 64 Tasten umgerechnet und für Zeichen- und Steuertasten getrennt

```
          REM *** Programm 5 *******************
505       REM DER VOLLSTÄNDIGE ASCII-CODE *
506       REM **********************************
507       PRINT CHR$(147)
510       FOR I=0 TO 255
520       IF I<0 THEN I=255
530       PRINT:PRINT:PRINT:PRINT:PRINT
540       PRINT"__________"
550       PRINT
560       PRINT I;"[" CHR$(I);"]";"... AAA"
570       PRINT
580       PRINT "__________"
590       Z=PEEK(203)
600

                IF Z=64 THEN 600
          610   IF Z=39 THEN POKE 36879,11:GOTO 600
          620   IF Z=47 THEN POKE 36879,27:GOTO 600
          630   IF Z=55 THEN POKE 36879,155: GOTO 600
VC 20     640   IF Z=61 THEN I=I-2
          650

                IF Z=64 THEN 600
          610   IF Z=4 THEN POKE 53280,3: POKE 53281,0: GOTO 600
          620   IF Z=5 THEN POKE 53280,3: POKE 53281,1:GOTO 600
          630   IF Z=6 THEN POKE 53280,3: POKE 53281,10: GOTO 600
C 64      640   IF Z=43 THEN I=I-2
          650

          FOR T=1 TO 100:NEXT T
660       PRINT CHR$(147)
670       NEXT I
680       GOTO 520
690
```

Bild 4: Programm zur Bestimmung des Kompletten ASCII-Codes

Steuercodes

in die Speicherzellen 203 und 653 gebracht werden.

Mit dem folgenden kleinen Programm haben wir dann diese beiden Speicherzellen abgefragt und eine Tabelle angefertigt.
Tippen Sie ein:
100 PRINT PEEK(203),PEEK(653)
200 GOTO 100

Für diejenigen Leser, die Teil 1 nicht gelesen oder keine Tabelle angefertigt haben, habe ich diesmal die Tabelle dabei. Der Grund, daß die Zahlen für die beiden Computer, trotz gleicher Tastatur, verschieden sind, liegt darin, daß die elektrische Anordnung der Tasten, in einer 8 x 8-Matrix (die ich das letzte Mal für den VC 20 gezeigt habe), beim C 64 anders sind.

```
5 REM******C-64  PROGRAMM 1**********
6 REM TASTENABFRAGE IN 203 UND 653 *
7 REM*******************************
10 PRINT CHR$(147)
20 A=PEEK(203)
30 B=PEEK(653)
40 IF A=4 AND B=0 THEN POKE 53280,6: POKE53281,2
50 IF A=4 AND B=1 THEN POKE 53280,5: POKE 53281,2
60 IF A=5 AND B=4 THEN POKE 53280,1: POKE 53281,1
70 IF A=46 AND B=0 THEN POKE 53280,3: POKE 53281,1
80 GOTO 20

105 REM****C 64 PROGRAMM 2 **********
106 REM UMRECHNUNG DES TASTEN-CODES *
107 REM          IN ASCII-CODE        *
108 REM*******************************
110 A=PEEK(203)
120 B=PEEK(653)
130 IF B=0 THEN Z=60289
140 IF B=1 THEN Z=60354
150 IF B=2 THEN Z=60419
160 IF B=4 THEN Z=64632
170 C=PEEK(Z+A)
200 PRINT A;B;C
210 FOR T=1 TO 200:NEXT
220 GOTO 110

305 REM****C-64 PROGRAM 3 ***********
306 REM   ABFRAGE DES ASCII-CODES   *
307 REM   AUS DEM TASTATURPUFFER    *
308 REM*******************************
310 PRINT CHR$(147)
320 POKE 198,0
330 A=PEEK(631)
340 PRINT A
350 GOTO 320
```

Bild 3: Die im Text erklärten Programme für den C 64

```
5 REM***VC-20    PROGRAMM 1**********
6 REM TASTENABFRAGE IN 203 UND 653 *
7 REM*******************************
10 PRINT CHR$(147)
20 A=PEEK(203)
30 B=PEEK(653)
40 IF A=39 AND B=0 THEN POKE 36879,126
50 IF A=39 AND B=1 THEN POKE 36879,45
60 IF A=47 AND B=4 THEN POKE 36879,25
70 IF A=53 AND B=0 THEN POKE 36879,27
80 GOTO 20

105 REM*** VC-20 PROGRAMM 2 **********
106 REM UMRECHNUNG DES TASTEN-CODES *
107 REM          IN ASCII-CODE        *
108 REM*******************************
110 A=PEEK(203)
120 B=PEEK(653)
130 IF B=0 THEN Z=60510
140 IF B=1 THEN Z=60575
150 IF B=2 THEN Z=60640
160 IF B=4 THEN Z=60835
170 C=PEEK(Z+A)
200 PRINT A;B;C
210 FOR T=1 TO 200:NEXT
220 GOTO 110

305 REM***VC-20 PROGRAM 3 ***********
306 REM   ABFRAGE DES ASCII-CODES   *
307 REM   AUS DEM TASTATURPUFFER    *
308 REM*******************************
310 PRINT CHR$(147)
320 POKE 198,0
330 A=PEEK(631)
340 PRINT A
350 GOTO 320
READY.
```

Bild 2: Die im Text erklärten Programme für den VC 20

TASTE	VC-20		C-64	
	203	653	203	653
nichts	64	0	64	0
f-1	39	0	4	0
f-3	47	0	5	0
f-5	55	0	6	0
f-7	63	0	3	0
A	17	0	10	0
B	35	0	28	0
C	34	0	20	0
D	18	0	18	0
E	49	0	14	0
F	42	0	21	0
G	19	0	26	0
H	43	0	29	0
I	12	0	33	0
J	20	0	34	0
K	44	0	37	0
L	21	0	42	0
M	36	0	36	0
N	28	0	39	0
O	52	0	38	0
P	13	0	41	0
Q	48	0	62	0
R	10	0	17	0
S	41	0	13	0
T	50	0	22	0
U	51	0	30	0
V	27	0	31	0
W	9	0	9	0
X	26	0	23	0
Y	11	0	25	0
Z	33	0	12	0
1	0	0	56	0
2	56	0	59	0
3	1	0	8	0
4	57	0	11	0
5	2	0	16	0
6	58	0	19	0
7	3	0	24	0
8	59	0	27	0
9	4	0	32	0
0	60	0	35	0
+	5	0	40	0
-	61	0	43	0
*	14	0	49	0
/	30	0	55	0
=	46	0	53	0
↑	54	0	54	0
←	8	0	57	0
.	37	0	44	0
:	45	0	45	0
,	29	0	47	0
;	22	0	50	0
£	6	0	48	0
@	53	0	46	0
CRSR←	23	0	2	0
CRSR↑	31	0	7	0
DEL	7	0	0	0
HOME	62	0	51	0
STOP	24	0	63	0
RETURN	15	0	1	0
SPACE	32	0	60	0
SHIFT	64	1	64	1
C=	64	2	64	2
CTRL	64	4	64	4
SHIFT u. C=	64	3	64	3
SHIFT u.CTRL	64	5	64	5
C= u. CTRL	64	6	64	6
SHIFT u. C= u.CTRL	64	7	64	7

Bild 1: Tabelle des Tastencodes in den Register 203 und 653

Im Gedenken an diesen Spruch meines Latein-Lehrers zeige ich noch einmal die Anwendung dieser Code-Zahlen in dem folgendem Programm 1. Zeilen in (), wie gesagt, sind für den C 64, aber nur dort, wo er sich vom VC 20 unterscheidet.

Die Wiederholung ist die Mutter der Weisheit

Natürlich wähle ich wieder die Funktionstasten und am besten auch noch eine andere Tastenkombination. Zweck der kleinen Demonstration soll das Umschalten auf verschiedene Bildschirmrahmen- und Hintergrundfarben sein. Ich schlage vor, Sie nehmen wie üblich die Zeitschrift zum Computer und lesen tippend weiter. Es folgt nun das Programm 1 zur Tastaturabfrage.
10 PRINT CHR$(147)

Diese Befehlsfolge löscht den Bildschirm. Die Code-Zahlen der Funktion CHR$ werde ich noch erklären.

Die Zeilen 20 und 30 erleichtern die Tipperei und machen übrigens das Programm ein bißchen schneller. Sie ordnen den Variablen A und B den Inhalt der Speicherzellen 203 und 653 — die wir ja abfragen wollen — zu.
20 A=PEEK(203)
30 B=PEEK(653)

Jetzt geht's los mit der Fragerei. Die Taste f1, mit der wir die Farbkombination Blau/Gelb schalten wollen, hat folgende Code-Zahlen:
40 IF A=39 AND B=0 THEN POKE 36879,126
(40 IF A=4 AND B=0 THEN POKE 53280,6:POKE 53281,7)
f2 ist dieselbe Taste, aber geSHIFTet (B=1). Die Farben sollen jetzt Rot/Grün sein:
50 IF A=39 AND B=1 THEN POKE 36879,45
(50 IF A=4 AND B=1 THEN POKE 53280,5: POKE 53281,2)
60 IF A=47 AND B=4 THEN POKE 36879,25
(60 IF A=5 AND B=4 THEN POKE 53280,1: POKE 53281,1)

Zeile 60 bestimmt ebenfalls eine Funktionstaste und zwar f3. Allerdings habe ich sie willkürlich mit der CTRL-Taste kombiniert, nicht, um Sie zu verwirren, sondern um zu zeigen, daß wir mit den f-Tasten mehr als acht Funktionen festlegen können, nämlich 32! (Vier f-Tasten mal acht Steuertasten-Kombinationen).'

Mit dem Klammeraffen »@« schalten wir die Farben in den Normalzustand zurück.
70 IF A=53 AND B=0 THEN POKE 36879,27
(70 IF A=46 AND B=0 THEN POKE 53280,3: POKE 53281,1)

Zeile 80 läßt das Programm im Kreis laufen, so daß die Funktionstasten beliebig oft ausprobiert werden können. S 80 GOT020

Funktionstasten im Überfluß

So, jetzt kommt der absolute Hit dieser Methode!

Ich habe gerade vorher gesagt, daß wir nicht acht, sondern 32 mögliche Funktionstasten haben, nämlich durch Verwendung der vier f-Tasten mit den acht Steuertasten-Codes in Speicherzellen 653.

Dasselbe gilt für jede andere Taste natürlich auch!

Der Computer benützt allerdings einige davon, zum Beispiel die 1 (SHIFT) für die Zeichen über den Zahlen beziehungweise rechts unten auf den Tasten, die 2 (C=) für die Zeichen links unten auf den Tasten und die 4 (CTRL) für die Farben.

Wir können daher mit den vom Computer benutzten Zahlen 3, 5, 6 und 7 aus Zelle 653 in Kombination mit allen 55 Tasten eine riesige Anzahl verschiedener »Funktionstasten« erfinden und sie in unseren Programmen einsetzen.

Der Computer geht bei der Abfrage noch einen Schritt weiter

Aber auch die 4 der CTRL-Taste, die ja nur die obere Reihe der Tasten beeinflußt, hat einen praktischen Wert, meiner Meinung nach sogar einen sehr großen, da sie ja nur einen einzigen Tastendruck erfordert und nicht eine Kombination.

Die CTRL-Taste kombiniert mit den 35 Tasten der unteren drei Reihen der Tastatur gibt uns mehr »Funktionstasten« als wir wahrscheinlich jemals brauchen werden.

Mehrere handelsübliche Zusatzmodule und -programme verwenden diese Methode, zum Beispiel auch die »Programmierhilfe« von Commodore. Verwenden Sie's doch auch! Das Kochrezept dazu steht oben in den Zeilen 50 bis 80.

Um das Ergebnis eines Tastendrucks weiter zu verarbeiten, wäre dem Computer die Abfrage zweier Speicherzellen zu langsam. Außerdem entsprechen diese Tastencodes keiner internationalen Norm, was in Verbindung mit anderen Geräten sehr lästig wäre.

Es gibt den international anerkannten ASCII-Code (American Standard Code for Information Interchange), der ursprünglich für die Zeichenübertragung von Fernschreibern erfunden wurde. Er besteht aus Dualzahlen mit einer Länge von 8 Bit oder falls Sie Dualzahlen nicht kennen (eine erste Einführung ins Dualsystem steht im Grafikkurs) aus Zahlen von 0 bis 255.

In diesen ASCII-Code wandelt der Computer nun die oben verwendeten Codezahlen der Tasten um. Die Umwandlung ist denkbar einfach.

Umrechnung des Tastatur-Codes in den ASCII-Code

Im nicht löschbaren Speicher (ROM) des Betriebssystems stehen vier Tabellen mit Zahlen.

Wenn man nun die Code-Zahl einer Taste zur Anfangsadresse der Tabellen addiert, erhält man eine Adresse, in der die ASCII-Codezahl gespeichert ist. Einfach, nicht wahr?

Probieren geht über studieren. Die Tabelle der ungeSHIFTeten Zeichen, also der Zahlen und Großbuchstaben, beginnt beim VC 20 ab Speicherzeile 60510, beim C 64 ab 60289.

Nehmen wir das »G«, sein Tatencode aus der Tabelle ist 19 (26).

Zu 60510 (60289) dazugezählt ergibt das 60529 (60315). Nun wollen wir mal nachschauen, was in dieser Zelle steht.

Geben Sie »direkt«, (das heißt ohne Zeilenzahl) ein:
für VC 20: PRINT PEEK (60529)
für C 64: PRINT PEEK (60315)

Das Resultat ist 71. Ein Blick auf die ASCII-Liste des Handbuches (oder in jede andere ASCII-Tabelle) zeigt uns die Richtigkeit dieser Aktion. Der ASCII-Code des Zeichens »G« ist 71.

Um alle ASCII-Codes abfragen zu können, schreiben wir wieder ein kleines Programm.
110 A=PEEK (203)
120 Z=60510
(120 Z=60289)
170 C=PEEK(Z+A)

Bitte verwenden Sie meine Zeilennummern, ich möchte nämlich später noch andere Zeilen einschieben.

Diese drei Zeilen sollten Ihnen klar sein. Zur Erinnerung: 60510 ist die Anfangsadresse der Code-Tabelle beim VC 20. Die 64er müssen also ihre eigene Adresse (60289) nehmen.

Als Ergebnis wollen wir noch den Tastaturcode (Zeile 110) und den ASCII-Code (Zeile 170) nebeneinander ausdrucken:

```
200 PRINT A;C
220 GOTO 110
```

Damit die beiden Zahlenbänder nicht zu schnell laufen, fügen wir noch eine Verzögerungsschleife ein.

```
210 FOR T=1 TO 200: NEXT
```

Dieser Programmablauf, der mit RUN 110 gestartet wird, reagiert jetzt auf jeden Tastendruck, links mit dem Tastencode, rechts mit ASCII. Worauf er aber nicht reagiert, sind geSHIFTete Zeichen, solche mit der Commodore-Taste »C« und die Farben (mit CTRL).

Sie wissen, warum? Natürlich, denn wir fragen ja nur die Speicherzellen 203 ab und nicht zusätzlich auch 653.

Jetzt muß ich noch schnell erwähnen, daß auch der Computer diese Zellen abfragt.

Wo steht was?

Erinnern Sie sich, ich habe oben gesagt, daß im ROM Tabellen stehen:

(1) ab 60510 (60289) für normale Zeichen
(2) ab 60575 (60354) für Zeichen mit SHIFT
(3) ab 60640 (60419) für Zeichen mit C=
(4) ab 60835 (64632) für Zeichen (Farben) mit CTRL

Das bauen wir jetzt in das Programm ein:

Zeile 120 ändern wir ab. Sie fragt jetzt die Speicherzelle 653 nach den Steuertasten ab. In den Zeilen 130 bis 160 springen wir auf die vier Tabellenanfänge. Der Ausdruck in Zeile 200 schließlich wird mit der Codezahl aus 653 erweitert, nämlich B.

```
120 B=PEEK(653)
130 IF B=0 THEN Z=60510 (60289)
140 IF B=1 THEN Z=60757 (60354)
150 IF B=2 THEN Z=60640 (60419)
160 IF B=4 THEN Z=60835 (64632)
200 PRINT A;B;C
```

Die Reihenfolge der Zahlenbänder auf dem Bildschirm — entsprechend der Zeile 200 — ist jetzt von links Zeichentaste, Steuertaste, ASCII-Code.

Der ASCII-Code wird im Tastatur-Puffer abgelegt

Ehrlich gesagt, das Verfahren der Tastaturabfrage in dem vorherigen Programm ist immer noch recht kompliziert. Eine direkte Abfrage des ASCII-Codes einer Taste wäre viel besser.

Und in der Tat, der Computer bietet sie uns. Er bringt nämlich jeden ASCII-Wert in einen Speicher zur Zwischenlagerung, bis er von einem Programmschritt gebraucht wird.

Dieser Speicher heißt »Tastatur-Puffer« und liegt von Speicherzelle 631 bis einschließlich 640.

Es können also maximal zehn Werte gespeichert werden. Das erste Zeichen steht immer in 631, alle anderen werden der Reihe nach in die folgenden Zellen gebracht.

Als erstes wird das Zeichen aus 631 ausgelesen und alle anderen rücken nach.

Wenn das Programm die ASCII-Werte aus dem Tastaturpuffer auslesen kann, dann können wir das natürlich auch. Das folgende kleine Programm soll es beweisen.

```
310 PRINT CHR$(147)
330 A=PEEK(631)
```

Sie sehen, wir wollen ganz einfach in Zelle 631 des Tastaturpuffers nachschauen, welcher ASCII-Wert nach Drücken einer Taste dort steht (Zeile 330). In Zeile 340 drucken wir den Wert aus und springen dann zum PEEK-Befehl zurück.

```
340 PRINT A
350 GOTO 330
```

Wenn Sie dieses Programm mit RUN 310 laufen lassen, werden Sie merken, daß es nicht geht, ich will sagen; noch nicht geht. Den Grund dafür habe ich kurz vorher schon angedeutet.

Wo sind die Computer-Detektive? Haben Sie's gemerkt?

Nun, ich habe erklärt, daß die ASCII-Werte der gedrückten Tasten der Reihe nach im Tastaturpuffer gespeichert werden und dann, wenn ein Wert aus 631 ausgelesen wird, nachrücken. Das ist der springende Punkt: durch PEEKen lesen wir nicht aus, wir schauen nur nach!

Es gibt zwei Lösungen für dieses Problem:

1) Wir verwenden einen Befehl, der den Wert herausholt; das ist GET oder INPUT.

2) Wir gaukeln dem Computer vor, daß der Tastaturpuffer nur aus einer einzigen Speicherzelle besteht.

Die Methode 2 ist exotischer, deshalb zeige ich Sie Ihnen zuerst. Es gibt eine Speicherzelle 198. In dieser Zelle steht eine Zahl, die angibt, wieviele Zeichen im Tastaturpuffer Platz haben. Normalerweise steht da eine 10 (schauen Sie nach).

Diese Zelle kann mit kleineren Zahlen gePOKEt werden, auch mit einer 0. Die 0 löscht sozusagen den Puffer. Das machen wir jetzt in unserem Programm vor dem PEEKen:

```
320 POKE 198,0
350 GOTO 320
```

Jetzt läuft's.

Die Zeile 340 gibt uns also den ASCII-Code der gerade gedrückten Taste auf dem Bildschirm aus, zum Beispiel die Zahl 84 für das »T«, 163 für das »—« (T mit C=) und so weiter.

Unter Verwendung von GET würde unser Programm so aussehen:

```
310 PRINT CHR$(147)
320 GET A$
330 IF A$="" THEN 320
340 PRINT ASC(A$)
350 GOTO 320
```

Neu ist die Verwendung der Funktion ASC. Sie bildet den ASCII-Wert des Zeichens A$.

Für diejenigen, die es noch nicht kennen: Zeile 330 nach dem GET ist erforderlich, da der GET-Befehl nicht auf das Drücken einer Taste wartet, sondern gleich weiterläuft. Solange aber keine Taste gedrückt ist, geht die Schleife nach 320 zurück.

Mit INPUT geht's noch kürzer, nur ist die Bedienung etwas umständlicher.

```
310 PRINT CHR$(147)
320 INPUT A$
340 PRINT ASC(A$)
350 GOTO 320
```

Der Umstand liegt daran, daß INPUT im Gegensatz zu GET auf einen Tastendruck wartet, der zusätzlich mit RETURN abgeschlossen werden muß.

Nach diesen Erklärungen und Versuchen müßten Sie eigentlich in der Lage sein, Programm 1 so umzuschreiben, daß statt der Abfrage der Speicherzellen 203 und 653 der Tastaturpuffer abgefragt wird. Ich schlage Ihnen vor, daß Sie das jetzt selbst ausprobieren, sozusagen als Hausaufgabe. Die Lösung, das Programm Nummer 4, ist an anderer Stelle in dieser Ausgabe versteckt.

Das einzige, was ich Ihnen verraten will, wissen Sie eigentlich schon, nämlich, daß Sie sich die ASCII-

Werte für die f-Tasten und den Klammeraffen »@« besorgen müssen. Aber wozu haben Sie Programm 3 gleich in drei Versionen?

Ich hoffe, daß Sie nach dieser Übung einsehen, daß Sie eine vollständige Liste aller ASCII-Codezahlen und ihrer Bedeutung unbedingt brauchen. Halt, werden Sie jetzt sagen, die Liste steht ja in jedem Handbuch — sogar in dem von Commodore.

Das stimmt, nur sind die meisten Listen nicht komplett.

Erinnern Sie sich? Ich habe vorher mal gesagt, daß der ASCII-Code die Werte von 0 bis 255 hat. Diese werden von den Commodore-Computern in nicht immer ganz der Norm entsprechender Weise für alle möglichen Zeichen und Sonderfunktionen verwendet, wie zum Beispiel die Farben, die Sonderzeichen auf den Tasten, Zeichenumschaltung und so weiter.

Auch die Funktion »Bildschirm löschen und Cursor auf HOME-Position« (das heißt die CLR/HOME-Taste) ist dabei — mit dem Codewert 147. Merken Sie was? Schauen Sie mal die jeweiligen 10er-Zeilen der Programme bisher an!

Es lohnt sich also schon, alle Code-Werte und die dazugehörigen Zeichen und Funktionen anzusehen.

Ihnen die Liste einfach abzudrucken wäre zu simpel. Sie sollen ja durch Experimentieren Ihren Computer besser kennenlernen. Ich liefere Ihnen die Versuchsanordnung dazu.

Vorher aber brauchen wir noch ein Hilfsmittel, welches uns erlaubt, aus einem ASCII-Wert das Zeichen beziehungsweise die Funktion zu ermitteln. Es ist die Umkehrung der in Programm 3 verwendeten ASC-Funktion. Sie kommt ebenfalls aus Basic und heißt CHR$(x).

Dieser Befehl liefert uns das Zeichen oder die Funktion des ASCII-Codes x.

Der Befehl PRINT CHR$(x) bringt Zeichen auf den Bildschirm. (Mit dem Befehl PRINT# a,CHR$ wird das Zeichen an ein beliebiges, mit der Nummer a bezeichnetes Peripheriegerät gebracht. Doch das will ich hier nicht weiter verfolgen.)

Jetzt aber zurück zu dem Hilfsmittel

Ergänzen Sie bitte in Programm 3 die Zeile 340 auf:

340 PRINT A, CHR$(A)

Jetzt druckt das Programm nach Start mit RUN 310 neben dem ASCII-Code auch das Zeichen, welches natürlich mit der gedrückten Taste identisch ist, auf den Bildschirm. Funktionen kann man allerdings nicht ausdrucken, sondern nur ihre Auswirkungen feststellen.

Doch nun zum Kochrezept. Der entscheidende Teil steht in Zeile 570.

570 PRINT I;"["CHR$(I)"]";... "AAA"

I ist die ASCII-Codezahl, die in einer FOR-NEXT-Schleife von 0 bis 255 hochgezählt wird. Die beiden Klammern [und] stehen in Anführungszeichen, damit sie ausgedruckt werden. Zwischen ihnen soll das zum Wert I gehörige Zeichen stehen.

In den Fällen, wo der ASCII-Code nicht ein Zeichen, sondern eine Funktion bedeutet, bleibt die Klammer leer. Aber der Code wirkt sich durch die Form PRINT CHR$(I) auf die 3 As aus (die Punkte ... stellen drei Leertasten dar). Zum Beispiel erscheinen sie nach I = 28 in roter Farbe, bei I = 17 (Cursor Down) eine Zeile tiefer.

Das Hochzählen von I in Zeile 520 wollen wir aber ein bißchen beeinflussen, und das natürlich mit Drücken von Funktionstasten und solchen, die wir dazu verdonnern.

In Zeile 600 werden daher solche Tasten abgefragt, mit der »alten« Methode in Zelle 203. Das ist reine Willkür beziehungsweise Sentimentalität von mir, die »neue« Methode über Zelle 631 geht genauso gut.

Wenn in 203 eine 64 steht (keine Taste gedrückt), dann wartet das Programm durch Rücksprung auf die Zeile 600.

Falls wir die f1-Taste drücken, schaltet Zeile 620 den Hintergrund auf Schwarz und geht wieder in Wartestellung. Diese und die beiden anderen Farbumschaltungen mit f3 (Zeile 630) auf weiß und mit f5 auf hellorange (Zeile 640) sind dann sehr nützlich, wenn die Farbe der drei As gegen den Hintergrund nur schlecht oder überhaupt nicht lesbar sind.

Zeile 650 gibt uns die Möglichkeit, mit der Minus-Taste in der Liste um 1 zurückzuschalten. Zeile 530 erlaubt ein Zurückschalten über die 0 nach 25. Erst wenn irgendeine beliebige Taste gedrückt wird, springt das Programm auf Zeile 660, wo nach kurzer Zeitverzögerung der Bildschirm gelöscht (70) und I weitergezählt wird (680 und 690).

So können Sie sich bequem alle 256 Werte des ASCII-Codes und ihre Wirkung anschauen.

Ich möchte Sie hier noch auf folgende Codewerte aufmerksam machen, deren Funktion Sie in diesem Programm entweder nicht sehen könnnen oder deren Funktion den Ablauf stören:

Code	Funktionen
3	entspricht der ungeSHIFTeten STOP/RUN-Taste
8	setzt die Umschaltung (mit SHIFT und C=) auf den 2. Zeichensatz außer Betrieb (ausprobieren!)
9	hebt die Sperre wieder auf
13	entspricht der RETURN-Taste
14	schaltet den 2. Zeichensatz per Programm ein (ich empfehle, danach wieder auf den »normalen« Zeichensatz zurückzuschalten)
131	entspricht LOAD/RUN (geSHIFTete STOP/RUN-Taste)
133-140	Funktionstasten f1 bis f8
141	geSHIFTete RETURN-Taste
142	schaltet den 1. Zeichensatz ein (Umkehrung von 14)
146	REVERSE-OFF (Taste 0 mit CTRL)
160	geSHIFTete SPACE-Taste (ja, ja, das gibt es auch!)

Ich empfehle Ihnen, mit diesem Programm 5 zu experimentieren. Versuchen Sie, besonders die Funktionen zu identifizieren, es ist nicht schwer. Zusätzlich sollten Sie alle sinnvollen ASCII-Werte mit den Ihnen zur Verfügung stehenden ASCII-Listen vergleichen. Verbessern und vervollständigen Sie diese Listen. Sie gehören zu Ihrem wichtigsten Handwerkszeug.

Im nächsten und zugleich letzten Teil werde ich Ihnen die Zusammenhänge zwischen dem ASCII-Code, dem Bildschirm-Code und dem PRINTen in Anführungszeichen erklären, natürlich wieder mit Kochrezepten.

Und schließlich will ich mein Versprechen einlösen, endlich die Methode der Abfrage von mehreren gleichzeitig gedrückten Tasten zu zeigen.

Übrigens, wenn Sie Fragen haben, scheuen Sie sich nicht, an den Verlag beziehungsweise über den Verlag an mich zu schreiben. Ich versuche mein Bestes.

(Dr. Helmut Hauck)

Reise durch die Wunderwelt der Grafik

Sie lernen in diesem Kurs nicht nur etwas über die Grafik. Ausführlich erläutert werden auch die beiden wichtigsten Zahlensysteme für den Computer, das Binär- und das Hexadezimalsystem. Wir ermöglichen Ihnen Änderungen an der Speicherorganisation und bringen Ihnen die logischen Verknüpfungen näher. Und das alles, um schließlich eigene Zeichen erstellen zu können.

Die zweite Etappe unseres Weges durch das Bytegewirr zu unserem Dornröschen (der hochauflösenden Grafik) wird gleich gestartet. Wir sollten uns nochmal in Erinnerung rufen, was wir bisher gesehen haben. Da war zunächst mal ein Überblick über die gesamte Speicherorganisation unseres Computers. Genauer haben wir uns dann die Ein- und Ausgabebausteine angesehen, um schließlich einen Plan der VIC-II-Chip-Register zu finden. Wir haben das Rätsel teilweise gelöst, wie ein bestimmtes Zeichen an einen bestimmten Ort des Bildschirmes gelangt und woher unser Computer überhaupt weiß, wie beispielsweise das A aussehen soll. Dabei sind wir bereits allerlei Merkwürdigkeiten begegnet: Wir gehen durch Alices Wunderland! Nun, der Wunder sind's noch nicht genug gewesen, denn auch auf dieser Etappe werden wir allerlei Eigenartigkeiten sehen: Wir treffen die Zweifingerlinge und die Sechzehnfingerlinge. Wir werden lernen, wie wir unseren Computer hinters Licht führen können. Schließlich werden wir uns wie Frankenstein — aber besser als er — an neue Kreationen heranwagen. Die Pause ist beendet, wir brechen auf.

Im Grunde genommen haben sie uns schon fast die ganze 1. Folge über ungesehen begleitet: Die Zweifingerlinge. Um sie für uns sichtbar zu machen, bedarf es eigentlich nur einiger Gedankenübungen. Beobachten Sie mal kleine Kinder beim Zählen oder Rechnen: Das läuft Finger für Finger. Wir haben zehn davon (im allgemeinen) und haben deswegen wohl auch neun Ziffern und die Null:

1, 2, 3, 4, 5, 6, 7, 8, 9, 0.

Um eine Zahl auszudrücken, die größer als 9 ist, zum Beispiel 9 + 1, setzen wir einfach zwei von diesen Ziffern zusammen und fangen wir wieder bei der kleinsten Ziffer 1 an und hängen eine Null dran: 10. Auf diese Weise können wir jeder Anzahl von Dingen eine Zahl zuordnen. Was wäre, wenn wir nur zwei Finger hätten? Wir hätten dann — wie die im Computer herumwimmelnden Zweifingerlinge — nur zwei Ziffern: 1 und 0.

Die Begegnung mit den Zweifingerlingen: Das Binärsystem

Um nun eine Zahl auszudrücken, die größer als unsere größte Ziffer (1) ist, würden wir auch so verfahren wie die Zehnfingerwesen. Also fangen wir wieder bei der kleinsten Ziffer an (die hier auch gleichzeitig die größte und überhaupt die einzige ist), also der 1 und hängen eine Null dran: 10. Wir zählen also jetzt

1, 10, 11, 100, 101, 110, 111, 1000, 1001 und so weiter.

Die Zehnfingerlingzahlen dafür sind: 1, 2, 3, 4, 5, 6, 7, 8, 9 und so weiter.

Die Zweifingerlinge würden also zu meinem guten alten R4 sagen: »Dieser R100 hat 100 Zylinder und 100010 PS«. Leider — oder von der Steuer her Gottseidank — hat sich dadurch aber an der Tatsache nichts geändert, daß er genauso schwach den Berg hinaufklettert wie vorher, nur das Zahlensystem, das ist jetzt Binär. Sehen wir uns das nochmal genauer an. Wissen Sie noch, was in der Mathematik Potenzen sind? Falls nicht, 10^3 heißt 10 mal 10 mal 10, also die Zehn dreimal mit sich selbst multipliziert. 10^0 ist allerdings 1. Wenn wir nun eine normale Dezimalzahl (eine Zahl der Zehnfingerlinge) vor uns haben, zum Beispiel 255, dann kann man dafür auch schreiben:

$255 = 2*10^2 + 3*10^1 + 5*10^0 = 2*100 + 5*10 + 5*1$

Rechnen Sie nach: Es stimmt! Genauso ist nun auch eine Binärzahl aufgebaut:

$1001 = 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0$

Deswegen ist es auch relativ einfach, die Zahlen der Zweifingerlinge in unser Zehnfinger-System umzurechnen:

$1001 = 1*2^3 + 0*2^2 + 0*2^1 + 1*2^0 = 1*8 + 0*4 + 0*2 + 1*1 = 9$

Die bequemste Methode ist es, ein Schema wie in Bild 1 zu benutzen. Andersherum kann man auch ganz einfach unsere Zahlen in die der Zweifingerlinge umrechnen, nämlich wie in Bild 2 gezeigt.

Dabei bedeutet lsb »least significant bit« und msb »most significant bit«, also zu deutsch etwa »bedeutsamstes Bit« und »am wenigsten bedeutsames Bit«. Das ist leicht zu verstehen: Es macht keinen so großen Unterschied, ob mir jemand 1001 Mark oder 1002 Mark schenkt. Der Unterschied berührt mich aber schon ganz anders bei 1001 Mark oder 2001 Mark. Für ökonomisch Denkende sei noch bemerkt, daß das Programm SpeiLu (in erweiterter Form hier angefügt) ein schönes Unterprogramm (Zeilen 20000 bis 20030) enthält, welches beliebige

Dezimalzahlen in Binärzahlen umrechnet.

Vielleicht haben Sie Lust, zur Übung noch ein bißchen zu rechnen (aber bitte nicht mit dem Unterprogramm):
a) Umrechnung dez→ binär 25,16,47,128
b) Umrechnung binär→ dez 10001,1110,11110000

Die Lösungen finden Sie am Ende dieser Folge.

Wir haben jetzt die Zweifingerlinge ausgiebig kennengelernt und werden mit ihrer Hilfe später einige Hürden nehmen können. Jetzt aber zu den Sechzehnfingerlingen.

Die Entartung der Sechzehnfingerlinge: Das Hexadezimalsystem

Diese Sechzehnfingerlinge begleiten uns auch ständig und zwar als Sechzehnfingerlinge getarnt. Doch bevor wir sie enttarnen, müssen wir etwas mehr über sie wissen. Die Zählweise der Sechzehnfingerlinge mutet uns, die wir nur zehn Finger haben (also auch nur 9 Ziffern und die Null) etwas merkwürdig an, denn was nimmt man — für Ziffern größer als 9 — ohne eine zweistellige Zahl zu benutzen? Wir bemühen das Alphabet:
0,1,2,3,4,5,6,7,8,9, A, B, C, D, E, F
Als Zehnfingerlinge würden wir dafür schreiben:
0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15
Sobald es um eine Anzahl größer als F (dez. = 15) geht, zum Beispiel F + 1, dann kommt auch hier die zweite Stelle dazu: F + 1 = 10 (dezimal wäre das: 15 + 1 = 16). Ebenso wie die Dezimalzahlen sind auch die Hexadezimalzahlen auf Potenzen aufgebaut:
$831 = $8*16^2+3*16^1+1*16^0=8*256+3*16+1*1=2097$

Dabei steht das Dollarzeichen dafür, daß es sich um eine Hexadezimalzahl handelt, was allgemein üblich ist. Die Umrechnung von Hex-Zahlen in Dezimalzahlen und umgekehrt ist etwas beschwerlicher als die von Binärzahlen. Man behilft sich am besten mit der Tabelle 1. Damit geht die Umrechnung einer Hex-Zahl in eine Dezimalzahl relativ einfach (siehe Bild 3).

Man sucht sich aus der Tabelle in der Spalte, die dem Stellenwert der Hex-Ziffer entspricht, die dazu gehörige Dezimalzahl. Das führt man für alle Hex-Stellen durch und addiert dann die so gefundenen Dezimalwerte wie im Beispiel gezeigt wird.

Bit	7	6	5	4	3	2	1	0	
binär	2^7 = 128	2^6 = 64	2^5 = 32	2^4 = 16	2^3 = 8	2^2 = 4	2^1 = 2	2^0 = 1	dezimal
11111111	1	1	1	1	1	1	1	1	255
10000111	1	0	0	0	0	1	1	1	135

Bild 1. Schema zur Umrechnung des Dual- in das Dezimalsystem

Auch die Umwandlung einer Dezimalzahl in die Hex-Zahl ist jetzt nicht mehr so tragisch. Nehmen wir die Dezimalzahl 40959 (siehe Bild 4). Wir suchen aus der Tabelle die größte Dezimalzahl heraus, die gerade noch kleiner als unsere Zahl ist: 36864. Ihr entspricht eine $9 an der 3. Stelle der zu findenden Hexzahl. Wir ziehen dann diese Zahl 36864 von unseren 40959 ab. Es bleiben 4095. Wieder suchen wir jetzt eine Stelle tiefer (also in Spalte 2) die größte Dezimalzahl heraus. Jetzt ist das 3840. Erneut folgt eine Subtraktion und so weiter wie im Beispiel im Bild 4 gezeigt wurde.

Ich bewundere Ihren Mut, daß Sie mit mir auf dieser Etappe soweit mitgegangen sind. Wir stecken jetzt anscheinend total fest im Dornendickicht. Aber nur noch eine letzte Anstrengung und wir kommen zu einer kleinen Lichtung, auf der wir uns etwas ausruhen können. Dazu werden wir nun die Sechzehnfingerlinge enttarnen.

Sehen wir uns dazu die Zahl $FFFF an, die größte mit vier Stellen darstellbare Hex-Zahl. Wenn Sie per Tabelle umrechnen, werden Sie feststellen, daß wir 65535 vor uns haben. Erinnern Sie sich an die Folge 1, wo diese Zahl die Obergrenze unseres gesamten Speichers war? Dann erinnern Sie sich sicherlich auch noch daran, daß in einer Speicher-Hausnummer (Byte) acht Zimmer (Bits) sind, die entweder leer oder voll sein konnten (0 und 1).

Bild 2. Schema zur Umrechnung des Dezimal- in das Dualsystem

```
175 : 2 = 87, Rest 1 → 1 ← lsb
 87 : 2 = 43, Rest 1 → 1
 43 : 2 = 21, Rest 1 → 1
 21 : 2 = 10, Rest 1 → 1
 10 : 2 =  5, Rest 0 → 0
  5 : 2 =  2, Rest 1 → 1
  2 : 2 =  1, Rest 0 → 0
  1 : 2 =  0, Rest 1 → 1 ← msb
Ergebnis: 1010 1111
          msb     lsb
```

In Bild 1 ist die Bit-Numerierung zu sehen und wie voll man so eine Hausnummer machen kann. Wie man aber auch dabei erkennt, ist die größte Zahl, die in einem Byte Platz findet 1111 1111 oder dezimal 255 oder — rechnen Sie nach — $FF im Hex-System. Nun kann man sich ja vorstellen, daß zum Beispiel das Betriebssystem häufig im Verlauf der Benutzung irgendwelche Hausnummern angeben muß, zum Beispiel wo eine Routine zu finden ist, ein Text und so weiter. Aber auch das Betriebssystem ist darauf festgelegt, daß in jeder Hausnummer nur 8 Bits vorhanden sind! Wie kann es dann aber zum Beispiel sagen, daß die Basic-Warmstartadressen bei 42115 liegt, wenn es nur bis 255 zählen kann? Ganz einfach: Es gibt sozusagen Wohngemeinschaften, die zwei Hausnummern bewohnen

```
        3 2 1 0
z.B. $  A 4 8 3
                    3
                  128
                 1024
                40960
               ------
              = 42115
```

Bild 3. Umrechnung einer Hex- in eine Dezimalzahl

und wenn alle Bits von 2 Bytes voll sind, dann haben wir: 1111 1111 1111 1111 = dezimal 65535. Wenn alle leer sind, haben

Bild 4. Umrechnung einer Dezimal- in eine Hex-Zahl

wir 0. Mit 2-Byte-Adressen kann also der ganze Bereich erfaßt werden. In Bild 3 haben wir berechnet, daß der Adresse 42115 die Hex-Zahl $A483 entspricht. Jetzt zerteilen wir diese Hex-Zahl auf zwei Bytes, von denen das eine MSB (most significant Byte = bedeutsamstes Byte) und das andere LSB (least significant Byte = am wenigsten bedeutsames Byte), genannt wird, wie vorhin msb und lsb bei den Bits.

MSB← →A4
LSB← →83

Beide sind kleiner als $F und können deswegen im Speicher untergebracht werden. Das Betriebssystem notiert sie sich in den Hausnummern 770 und 771 auf page 3. Sehen wir doch einfach mal nach! Geben Sie ein:

PRINT PEEK (770), PEEK (771)
»RETURN«

Wir erhalten: 131 164

Lassen Sie sich nicht verwirren! Rechnen wir diese Angaben mal um in Hex-Zahlen (Bild 5). Wir finden also die im Bild 5 dargestellten Werte.

Bild 5. Umrechnung von 131 (LSB) und 164 (MSB) in Hexzahlen

Das sieht alles komplizierter aus als es ist. Mit etwas Übung, die wir uns jetzt zulegen wollen, werden Sie feststellen, daß Sie allerhand damit anfangen können. Die Sechzehnfingerlinge sind also als Dezimalzahlen getarnt gewesen (Bild 6). Auch hier zur Übung einige Aufgaben:

c) Umrechnung hex → dez
$92, $D728, $A001

d) Umrechnung dez → hex
65534, 2048, 21235

e) Tun Sie so, als müßten Sie diese letzte Zahl in die Speicherzellen 770 und 771 eingeben. Welche POKEs sind nötig?

So, jetzt wo wir die Hex-Zahlen erkennen können, werden wir uns ihrer kräftig bedienen.

Speicher:	770	771
dezimal:	131	164
hex:	$83	$A4
also	LSB	MSB
$A483 = dez. 42115		

Bild 6. Inhalt der Speicherstellen 770 und 771

Wir führen den C 64 hinters Licht: Eigene Änderungen an der Speicherorganisation

Jetzt können wir geistig etwas ausspannen. Falls Sie Ihren Computer schon in Betrieb haben, speichern Sie darauf befindliche Programme ab, schalten Sie aus und wieder ein. Wir wollen den Computer im Ursprungszustand etwas untersuchen und dann einige Änderungen vornehmen. Auf der Zeropage gibt es einige nützliche Hausnummern, die wir uns ansehen wollen. Tippen Sie doch mal ein:

PRINT PEEK (43), PEEK (44)
»RETURN«

Wir erhalten 1 8

Die Umrechnung mit der Tabelle ergibt $ 801 = dez. 2049. Sehen wir in das Handbuch, Anhang Q auf Seite 160. Dort ist zu lesen, hier sei die Startadresse vom Basic-Text gespeichert. Jetzt machen wir uns das etwas komfortabler. Wir geben im Direktmodus (also ohne Programmzeilennummer) ein:

A = 45:PRINT PEEK(A), PEEK(A + 1), PEEK(A) + PEEK(A + 1)*256
»RETURN«

Wir erhalten: 3 8 2051

Dies ist die Adresse, von der an Variable gespeichert werden. Gleichzeitig erfährt man so, wo ein Basic-Programm aufhört, denn die einfachen Variablen werden direkt hinter dem Basic-Programmtext gespeichert. Jetzt fahren wir den Cursor hoch auf die 45 in der zuletzt eingegebenen Zeile und ändern sie um auf 47, dann »RETURN«. Es erscheint

10 8 2058

Ab 2058 beginnen jetzt die indizierten Variablen. Normalerweise fangen sie im Leerzustand auch bei 2051 an. Wir haben aber eine Variable A definiert und die verschiebt die indizierten Variablen um 7 Bytes. Als Nebeneffekt sehen wir so, daß eine Variable in 7 Bytes gelagert wird. Zur Kontrolle geben wir nochmal ein:

CLR:PRINT PEEK(47), PEEK(48) »RETURN«
und erhalten 3 8 na also!

Diese Untersuchung können Sie noch weiterführen, wenn Sie wollen. Sie erhalten so die Werte in Tabelle 2.

Bevor wir jetzt an die erste Änderung gehen, sehen wir mal nach, wieviel freies Basic-RAM wir zur Verfügung haben:
PRINT FRE(0) + 65536 »RETURN«
Es sollte auch bei Ihnen erscheinen: 38909.
Rechnen Sie entsprechend Bild 7 nach.
Nun wollen wir unserem Computer einreden, sein Basic-Speicher sei schon bei 12288 statt bei 40960 zu Ende. Zunächst müssen wir umrechnen. Wir sehen in die Tabelle 2 und rechnen entsprechend Bild 8 wie gehabt. Also ist das MSB = $30 und das LSB = $00. Jetzt rechnen wir wieder ins Dezimalsystem um:

3*16 = 48		0*16 = 0
0* 1 = 0	und	0*1 = 0
MSB = 48		LSB = 0

Die höchste Basic-Adresse ist (siehe Tabelle 2) in MEMSIZ gespeichert und deshalb geben wir ein:
POKE 55,0:POKE 56, 48 »RETURN«
Nun sehen wir nach mit
PRINT FRE(0) und erhalten 10237.

Es ist also geglückt. Der noch freie RAM-Bereich oberhalb von 12288 wird für Basic vom Computer nicht mehr wahrgenommen. Das nennt man »schützen« eines Speicherbereiches vor dem Überschreiben durch Basic. Gleichzeitig sollte man, falls im Basic-Programm auch Strings verwendet werden, auch noch FRETOP berücksichtigen mit:
POKE 51, 0:POKE 52, 48 »RETURN«

Der nun verfügbare Bereich ist in Bild 9 zu erkennen. In SpeiLu wird dieser Schutz in Zeile 10 vollzogen. Jedesmal, wenn man einen Teil des Basic-Speichers für andere Dinge verwenden will als für Basic, muß man diesen Teil in der gezeigten Weise schützen.

Sie werden vielleicht sagen, daß Sie soooo lange Basic-Programme kaum verwenden und nie in Regionen über 20000 oder 25000 geraten werden. Leider ist das ein Irrtum. Denn die Speicherung von Strings geschieht ab Adresse 40960 abwärts (siehe Bild 7). Ein »sicherer« Bereich im Basic-Speicher (ohne ihn schützen zu müssen) könnte höchstens irgendwo ungewiß mitten drin sein, wo weder von unten das Basic-Programm mit seinen Variablen noch von oben die Strings anstoßen würden. Darauf würde ich mich aber lieber nicht verlassen. Schützen ist besser. Wir werden im Verlauf der weiteren Folgen noch eine Reihe weiterer Möglichkeiten benutzen um die Speicherorganisation umzukrempeln.

Bild 7. Der freie Basicbereich mit den entsprechenden Zeigern


```
        AENDERUNGEN FUER     S P E I L U

READY.

140 PRINT:PRINT"      (3) AENDERN EINES ZEICHENS"
145 PRINT:PRINT"      (4) PROGRAMM-ENDE"

READY.

170 A=VAL(A$):IFA<0ORA>4THEN160
180 ONAGOSUB1000,2000,3000,4000
        ERGAENZUNG   FUER     S P E I L U

3000 PRINTCHR$(147);CHR$(18);TAB(10);"AENDERN VON ZEICHEN"
3010 PRINT:PRINT"  ES KOENNEN ZEICHEN GEANEDERT WERDEN"
3020 PRINT"INDEM DER 'BILDSCHIRM-CODE' DES ZEICHENS";
3030 PRINT"     EINGEGEBEN WIRD. ( 0 - 511)":IFTS=0THENGOSUB40000
3035 PRINT:PRINT"     ANDERN SIE DIE BINAERZAHL MIT"
3037 PRINT"'0' ODER '1' ENTSPRECHEND. DANN <RETURN>"
3040 PRINT:INPUT"CODE : ";A:IFA<0ORA>511THENRETURN
3050 PRINTCHR$(147):FORAD=12288+8*ATO12288+8*A+7
3060 DE=PEEK(AD):GOSUB10000:GOSUB20000:GOSUB30000:NEXTAD
3070 PRINTCHR$(19):AD=AD-9
3080 FORQ=1TO8:PRINTTAB(19);:INPUTA$:AD=AD+1:GOSUB50000:GOSUB10000:GOSUB20000
3090 PRINTCHR$(145);"                                      "
3100 PRINTCHR$(145);CHR$(145):GOSUB30000:POKEAD,DE:NEXTQ
3110 GETA$:IFA$=""THEN3110
3120 RETURN

READY.

50000 DE=0:FORI=1TO8:IFMID$(A$,I,1)="1"THENDE=DE+2^(8-I):
50010 NEXTI:RETURN

READY.
```

Änderungen von Speilu, um eigene Zeichen kreieren zu können

Und oder? Oder und? Die Befehle AND, OR

Jetzt haben wir beinahe alles Handwerkszeug beieinander, um unabhängig von irgendwelchen Fertigprogrammen uns selbst neue Zeichen zu definieren und auch zu bestimmen, woher der Computer sie dann holen soll. Nur eine Tatsache stört noch. Wir wissen jetzt zwar, wie wir in unserem C 64 ganze Bytes ändern können indem wir Adressen POKE-fertig umrechnen und dann einPOKEn. Was tun wir aber, wenn

Bild 9. Der geschützte Bereich von 12288 bis 40960

— was uns häufig beschäftigen wird — nicht das ganze Byte, sondern nur ein halbes (ein sogenannter Nibble) oder gar nur ein einziges Bit verändert werden soll? Natürlich gibt es dann fast immer die Möglichkeit, durch ein PEEK nachzusehen, was im Byte drin ist, das dann ins Binärsystem umzurechnen, dann die Binärzahl nach unserem Wunsch zu ändern, sie wieder ins Dezimalsystem umzurechnen und dann schließlich einzuPOKEn.

Sehr umständlich! Basic sei Dank gibt es da zwei Befehle, die uns den Aufwand verringern helfen: AND und OR. Es handelt sich um sogenannte logische Operatoren, die zwei Dinge oder Aussagen miteinander verbinden und daraus ein Ergebnis produzieren. Zunächst mal zu AND. Wir kennen das von Basic her zum Beispiel in der IF...THEN..Verzweigung:

5 IF A = 2 AND B = 200 THEN 10

Nur dann, wenn A = 2 und B = 200 ist, erfolgt ein Sprung nach Zeile 10, das heißt, wenn beide miteinander verknüpften Bedingungen erfüllt sind, ist das Ergebnis der Verzweigung erzielt. Bei binären Zahlen ist das einfacher:

1 AND 1 = 1. Wenn also beide Ziffern 1 sind, ist das Ergebnis 1. Man faßt das gerne in einer Tabelle zusammen (Tabelle 3).

Wie wendet man das an? Nehmen wir an, ein Byte sähe binär so aus: 1111 1011

	1111 1011	unser Byte
	0000 1111	eine sogenannte Maske
AND	————	
	0000 1111	unser Ergebnis

Halt! Geben Sie das aber nicht wirklich ein, denn damit verändern Sie den Basic-Warmstart-Vektor

Bild 8. 12288 dez. in eine Hex-Zahl umgewandelt

und den brauchen wir noch. Sollten Sie's schon getan haben, dann erfreuen Sie sich noch ein wenig des Effektes und ziehen Sie dann die Notbremse: Computer aus- und wieder anschalten.

Nun zu OR. Auch das kennen wir vom Basic her, zum Beispiel:

5 IF A = 2 OR B = 200 THEN 10

Wenn also A = 2 ist oder wenn B = 200 ist oder wenn beide Bedingungen erfüllt sind, erfolgt der Sprung nach 10. Ebenso wie für AND kann man auch hier die Verhältnisse am besten mit einer Tabelle übersehen (Tabelle 4).

Wir möchten es verändern, so daß es zu 0000 1011 wird. Dann setzen wir die AND-Operation ein: Das heißt, alle Bits, die mit einer 1 AND-verknüpft worden sind, bleiben unverändert. Alle Bits, die dagegen mit einer 0 AND-verknüpft wurden, sind jetzt 0. Anstelle der ganzen Rechnerei muß also jetzt nur die Maske umgerechnet werden: 0000 1111 = 15 dezimal. Nehmen wir an, unser Byte wäre die Adresse 770, dann sähe die Änderung jetzt so aus:

POKE 770, PEEK (770) AND 15

Die Anwendung sieht dann so aus (wir nehmen unser Ergebnis von vorhin 0000 1011):

	0000 1011	
	1111 0000	eine Maske
OR	————	
	1111 1011	das neue Ergebnis.

Überall dort also, wo mindestens eine 1 steht, ergibt sich im Endausdruck auch eine 1.

Während man mit AND gezielt Bits löschen kann, vermag man mit OR gezielt Bits zu setzen. Beide Operationen können natürlich auch miteinander kombiniert werden. Es gilt die alte Jungprogrammiererregel: Probieren, probieren,...

Ein Beispiel stelle ich Ihnen nochmal genau vor, das wir gleich verwenden werden. Erinnern Sie sich, daß wir in der letzten Folge das Byte 53272 etwas genauer angesehen haben. Die unteren 4 Bits (genau genommen ohne Bit 0) geben an, wo die Punktmuster für die Zeichen abrufbereit stehen. Durch PEEK (53272) fanden wir den Dezimalwert 21. Das entspricht dem Binärwert 0001[010]1.

Wobei der eingerahmte Teil also für den Ort der Zeichen zuständig ist. In der Tabelle 5 sehen Sie, welche Kombinationen auf welche Speicherorte als Startadressen unserer Zeichenmuster deuten.

Wenn wir nun also einen anderen Ort eingeben wollen, dürfen wir nur die Bits 1 bis 3 verändern. Lassen Sie uns die Zeichen nicht mehr von 4096 an, sondern von 6144 an gespeichert haben! Zunächst einmal müssen die Bits 4 bis 7 vor jeder Änderung geschützt sein und die Bits 0 bis 3 gelöscht werden:

dez. 21	0001 0101	das ist unser PEEK (53272)
dez. 240	1111 0000	eine Maske, die AND-verknüpft wird
dez. 16	0001 0000	das ist: PEEK (53272) AND 240

Jetzt können wir gezielt Bits setzen. Wir brauchen die Kombination 011X. Wenn wir für X einfach 0 an-

nehmen (das geht, weil Bit 0 hier nicht beachtet wird) dann ergibt das einen Dezimalwert von 6 (siehe Tabelle 5).

dez. 16	0001 0000	unser Zwischenwert
dez. 6	0000 0110	Maske wird OR verknüpft
dez. 22	0001 0110	unser Endwert

Bild 10. Der kopierte Zeichensatz im RAM

Alles in allem geben wir ein:
POKE 53272, (PEEK (53272) AND 240) OR 6
Das können Sie gefahrlos eingeben und sich am Ergebnis freuen. Wenn Sie danach übrigens mal mit PEEK (53272) abfragen, werden Sie nicht 22, sondern 23 erhalten, was an Bit 0 liegt, das wir so nicht beeinflussen können.

Frankensteins freundliches Monster: Eigene Zeichen

Wieso kann man eigentlich eigene Zeichen definieren, wo es sich doch um ein Zeichen-ROM handelt, woraus der C 64 seine Zeichen bezieht? Zum Umbauen der Zeichen muß man doch in die Punktmatrix hineinschreiben und das geht nur ins RAM. Na, dann kopieren wir doch einfach das Zeichen-ROM in den RAM-Bereich. Dort können wir dann nach Herzenslust herumPOKEn. Dazu muß man allerdings wissen, daß drei Dinge zu beachten sind:

a) Sobald wir dem Computer gesagt haben, wo er seine Zeichen herholen soll, kennt er alle die Zeichen nicht mehr, die nicht mit kopiert worden sind. Man muß sich also vorher überlegen, welche Zeichen man braucht und erst dann kopieren (siehe dazu in Folge 1 die Tabelle 2).

b) In Folge 1 ist erwähnt worden, daß der Computer so gebaut ist, daß er ständige Wechsel durchführt zwischen den Etagen unseres Speichers. Außerdem verrichtet er noch eine Reihe anderer Tätigkeiten nach einem schnell vor sich gehenden System von lauter Unterbrechungen. Beim Kopiervorgang sollte keine Unterbrechung stattfinden, weil sich der Computer solange auf das Zeichen-ROM konzentrieren soll. Man muß also das sogenannte Interrupt-System während des Kopierens abschalten.

c) Wir kopieren unsere Zeichen ins RAM, müssen den dafür verwendeten Speicherraum also vor dem Überschreiben durch ein Basic-Programm schützen.

Es empfiehlt sich folgende Vorgehensweise:

A. Schützen des RAMs. Dazu wollen wir den Bereich verwenden, der auch in SpeiLu eine Rolle spielt
10 POKE 52, 48: POKE 56, 48
Für das folgende sollten Sie sich das Unterprogramm ab Zeile 40000 von SpeiLu ansehen.

B. Abschalten des Interrupt. Das macht einen Besuch beim CIA # 1, Hausnummer 56 334 nötig. Mit einer AND-Operation knipsen wir die Unterbrechung ab:
20 POKE 56 334, PEEK (56 334) AND 254

C. Nachdem der Computer nicht mehr per Interrupt die Etagen abläuft, muß er behutsam zum Zeichen-ROM geführt werden:
30 POKE 1, PEEK (1) AND 251
Behutsam deswegen, weil wir Byte 1 aus der ersten Folge noch in unguter Erinnerung haben. Wir werden es später besser kennenlernen.

D. Nun steht dem Kopieren nichts mehr im Wege. Wir kopieren alles:
40 FOR I = 0 TO 4 095:POKE 12 288 + I, PEEK (53248 + I): NEXT
Das dauert allerdings eine Weile.

E. Nun muß das Interrupt-System wieder in den Ausgangszustand zurückversetzt werden:
50 POKE 1, PEEK(1)OR 4
60 POKE 56 334, PEEK(56 334)OR 1

F. Jetzt teilen wir dem Computer mit, daß er in Zukunft seine Zeichen ab 12 288 und nicht mehr im Zeichen-ROM findet:
70 POKE 53 272, (PEEK(53 272) AND 240) OR 12

Byte	binär		dezimal
12296	00011000	=	24
12297	00111100	=	60
12298	01100110	=	102
12299	01111110	=	126
12300	01100110	=	102
12301	01100110	=	102
12302	01100110	=	102
12303	00000000	=	0

Bild 11. Das Zeichen A jetzt im RAM

Byte	binär		dezimal
12296	01100110	=	102
12297	00000000	=	0
12298	00011000	=	24
12299	00011000	=	24
12300	10000001	=	129
12301	01100110	=	102
12302	00111100	=	60
12303	00000000	=	0

Bild 12. So soll unser neues »A« aussehen

Nach dem RUN merken Sie — wenn alles richtig war — noch keinen Unterschied, außer, daß wir uns eine Menge Speicherplatz weggeschnitten haben. Aber nun wollen wir ans Zeichenumbauen gehen. Nehmen wir mal an, daß wir den Buchstaben A zu langweilig finden.

Wir werden ihm ein neues Image verleihen. Wenn Sie sich an die Folge 1 erinnern, dann ist der Buchstabe A nach dem Klammeraffen @ der zweite Buchstabe. Sein erstes Byte ist an achter Stelle des Zeichen-ROMs, also ab 53 256, zu finden. Nachdem wir jetzt kopiert haben, finden wir ihn ab 12296 (siehe Bild 10), und jetzt verstehen wir auch das Bild 8 in Folge 1 als gesetzte und gelöschte Bits anzusehen (Bild 11).

Wir zeichnen uns ein 8 x 8-Raster und konstruieren darin unser neues »A« (Bild 12).

Dann berechnen wir die Dezimalwerte der Bytes und geben schließlich ein:

80 POKE12296, 102:POKE12297,0: POKE12298, 24:POKE12299,24: POKE12300, 129
90 POKE12301, 102:POKE12302,60: POKE12303, 0

Wenn Sie dieses Porgramm mit RUN starten, dann lächelt Sie künftig das A freundlicher an als bisher. Natürlich läßt sich das alles auch viel eleganter lösen. Besonders die Zeilen 80 und 90 können durch eine kleine Schleife, die DATA-Zeilen liest und in den Speicher POKEd, ersetzt werden. Die Adressen können durch Multiplikation des Commodore-Codes mit 8 berechnet werden:
Startadresse des Zeichens mit Code C = 12288 + 8*C

Außerdem finden Sie im Anschluß das erweiterte Programm SpeiLu, das es auf einfache Weise gestattet, Zeichen zu ändern. Nach dem Ändern der Zeichen kann auch der normale Zeichensatz wieder benutzt werden — wenn nötig — durch Eingabe des ursprünglichen Wertes 21 in Hausnummer 53272. Falls Sie SpeiLu benutzt haben, gibt es zwei Möglichkeiten, den alten Zustand wieder herzustellen:
1) Computer aus- und wieder anschalten oder
2) POKE 53272, 21:POKE52,160: POKE56, 160 (Warum, das wissen Sie ja jetzt aus dieser Folge)

Schließlich noch eine Bemerkung: Obwohl manche Basic-Befehle nach der Änderung etwas vergnügter aussehen als vorher, funktioniert zum Beispiel T?B(5) genausogut wie TAB(5). Testen Sie mal: TAB(10)"ABRAKADABRA".

Wenn Sie auf Kleinschreibung oder REVERSE umschalten, sieht alles wieder normal aus.

Damit sei's für heute genug. Es war eine schwere Etappe. Sie haben sich tapfer durch das Dornengestrüpp geschlagen! Sie werden es nicht bemerkt haben aber wir sind Dornröschen schon ziemlich nahe gekommen. Bis zum nächsten Aufbruch können Sie sich die Zeit damit vertreiben, einen eigenen Zeichensatz herzustellen.

(Heimo Ponnath)

Hier noch die Lösungen der Aufgaben:
a) 11001, 10000, 101111, 10000000
b) 17, 14, 240
c) 146, 55080, 40961
d) $FFFE, $800, $52F3
e) POKE 770, 243:POKE 771, 82

Adresse			Normaler Inhalt im Leerzustand			
Commodore Name (Label)	LSB	MSB	LSB	MSB	Dezimal	Bedeutung
TXTTAB	43	44	1	8	2049	Anfang Basic-Text
VARTAB	45	46	3	8	2051	Variablenstart
ARYTAB	47	48	3	8	2051	Arraystart
STREND	49	50	3	8	2051	Arrayende + 1
FRETOP	51	52	0	160	40960	Stringstart
MEMZIZ	55	56	0	160	40960	höchste Basic-Adresse
MEMSTR	641	642	0	8	2048	RAM-Start für Betriebssystem
MEMSIZ	643	644	0	160	40960	RAM-Ende für Betriebssystem

Tabelle 2. Die wichtigsten Adressen mit ihren Inhalten in der Zeropage

$	dez.	$	dez.	$	dez.	$	dez.
0	0	0	0	0	0	0	0
1	4096	1	256	1	16	1	1
2	8192	2	512	2	32	2	2
3	12288	3	768	3	48	3	3
4	16384	4	1024	4	64	4	4
5	20480	5	1280	5	80	5	5
6	24576	6	1536	6	96	6	6
7	28672	7	1792	7	112	7	7
8	32768	8	2048	8	128	8	8
9	36864	9	2304	9	144	9	9
A	40960	A	2560	A	160	A	10
B	45056	B	2816	B	176	B	11
C	49152	C	3072	C	192	C	12
D	53248	D	3328	D	208	D	13
E	57344	E	3584	E	224	E	14
F	61440	F	3840	F	240	F	15

Tabelle 1. Umrechnungstabelle von Hex in Dez.

AND	1	0
1	1	0
0	0	0

Tabelle 3. Die AND-Verknüpfung

OR	1	0
1	1	1
0	1	0

Tabelle 4. Die OR-Verknüpfung

Zahlenwert Bits 0-3 (Bit 0 = 0)	Bitmuster des Byte 5372	Startadresse Zeichenspeicher dez.	hex	
0	XXXX000X	0	$0000	
2	XXXX001X	2048	$0800	
4	XXXX010X	4096	$1000	Einschaltzustand
6	XXXX011X	6144	$1800	
8	XXXX100X	8192	$2000	
10	XXXX101X	10240	$2800	
12	XXXX110X	12288	$3000	
14	XXXX111X	14336	$3800	

Tabelle 5. Das Bitmuster und die Startadresse des Zeichenspeichers im Byte 53272

Strubs — ein Precompiler für Basic

In der letzten Ausgabe haben Sie den Unterschied zwischen einem Compiler und einem Interpreter erfahren und sich kurz über die Vorteile von Strubs informieren können. Hier nun werden die in Strubs implementierten Befehlsstrukturen erläutert und das Objektprogramm von Strubs selbst vorgestellt.

An anderer Stelle in dieser Zeitschrift (oder auch in den unten aufgeführten Büchern) können Sie sich ausführlich über die Grundlagen der strukturierten Programmierung informieren. Aus diesem Grund werden wir uns hier darauf beschränken, einige Aspekte kurz anzusprechen und im übrigen vorzustellen, was Strubs in dieser Hinsicht zu bieten hat.

Gehören Sie auch zu denjenigen, die sich manchmal ein Programm aus einer Zeitschrift vornehmen, um zu analysieren, wie es arbeitet oder um eventuell Teile des Programms für eigene Programmprojekte zu verwenden? Dann erinnern Sie sich bestimmt an Programme, bei denen Sie sich verzweifelt von Sprung zu Sprung bewegen und nach nicht allzu langer Zeit vollkommen den Überblick verlieren. Oder vielleicht kennen Sie folgende Situation: Sie schreiben ein Programm und erinnern sich angesichts eines bestimmten Problems, daß Sie ein ganz ähnliches Problem schon einmal in einem anderen Programm gelöst haben. Aber sobald Sie sich den alten Programmtext vornehmen, um den entsprechenden Programmteil in ihr neues Programm zu übernehmen, müssen Sie enttäuscht feststellen, daß diese spezielle Problemlösung so sehr in das Programmgeflecht verwoben ist, daß es Ihnen weitaus einfacher scheint, den entsprechenden Programmteil vollkommen neu zu entwickeln.

Die Ursache für solche Erscheinungen liegt zum Teil darin, daß viele Basic-Programme mehr oder weniger aus der Sicht des Computer der »Basic-Maschine« — direkt am Computer nach dem Verfahren von Versuch und Irrtum entwickelt werden. Das kann in Einzelfällen sogar soweit führen, daß man zum Schluß zwar sieht, daß das Programm läuft, aber selbst nicht so recht weiß, warum eigentlich und wie es funktioniert. Der Hauptgrund für solche Unübersichtlichkeit aber liegt in der Verwendung zahlreicher wilder Sprünge und ausgefallener Programmier-Tricks. (Daß die Verwendung von GOTO-Anweisungen den mathematischen Beweis für die Korrektheit von Programmen praktisch unmöglich macht, ist für den Informatiker interessant, braucht uns hier aber nicht zu interessieren).

Den entgegengesetzten Weg geht die strukturierte Programmierung. Sie bedeutet vor allem sorgfältige Planung und den Verzicht auf GOTOs und unübersichtliche Programmiertricks. Hier steht die systematische Analyse des Problems im Vordergrund. Die eigentliche Codierung, das heißt die Formulierung des Programmtextes in einer bestimmten Programmiersprache, spielt nur eine untergeordnete Rolle.

In der Problemanalyse geht es darum, ein gegebenes Problem in relativ selbständige Teilprobleme zu zerlegen und deren Beziehungen zueinander festzulegen. Den Aufbau des Programms Strubs mit den jeweiligen Zeilennummern können Sie Bild 1 entnehmen. Das komplette Objektprogramm ist ebenfalls abgedruckt (siehe Listing).

Entsprechend setzt sich das strukturierte Programm aus einer Reihe möglichst selbständiger Programmeinheiten zusammen. Dieses Vorgehen spiegelt sich im Konzept der Blöcke und Module.

Ein Block ist eine Anweisung oder eine Folge von Anweisungen mit genau einem Eingang und genau einem Ausgang. Das heißt man darf weder in einen solchen Block hineinspringen, noch aus diesem Block herausspringen. Solche Blöcke können entweder aneinander gereiht oder beliebig tief ineinander geschachtelt werden; sie dürfen sich

INITIALISIERUNG 45050
MENÜ 40050
EDITIEREN 8850
ÜBERSETZEN 5050
MARKENTABELLE AUSGEBEN 48050
FEHLERTABELLE AUSGEBEN 49050
1. LAUF 5550
2. LAUF 6550
MARKE DEFINIEREN 6050
BEFEHLE IM 1. LAUF 1550
BEARBEITE ZEILE 4050
SCHREIB-ZEILE 550
MARKE ÜBERSETZEN 1050
BEFEHLE IM 2. LAUF 2550

Bild 1. Aufbau von Strubs

ALLGEMEINE ROUTINEN:

NEXTCHAR	: 00250
HOLNAME	: 00750
ERROR	: 08050
ABBRUCH	: 50000
WARTEN	: 49550

-Programme (Teil 2)

aber nicht überschneiden. In letzterer Hinsicht verhält es sich mit diesen Blöcken also genauso, wie bei den bekannten FOR-Schleifen in Basic.

Ein strukturiertes Programm besteht nun ausschließlich aus einer geordneten Hierarchie solcher Blöcke. Der kleinste mögliche Block besteht aus einer einzelnen Anweisung, wie zum Beispiel PRINT "Text". Der größte, umfassendste Block besteht aus dem Programm selbst.

Da ist zunächst einmal die einfache IF-Anweisung, die schon von Basic her bekannt ist. Dieses normale Basic-IF kann natürlich wie alle Basic-Befehle weiterhin benutzt werden. Zusätzlich bietet Strubs aber eine erweiterte Form, bei welcher der THEN-Teil nicht auf den Rest einer Programmzeile begrenzt ist, sondern beliebig viele Zeilen umfassen kann, die durch den Befehl '!FI' — einfach ein umgedrehtes IF — abgeschlossen werden. Ein Beispiel:

```
10 ! IF X=Y THEN
20 :    PRINT "X und Y"
30 :    PRINT " SIND GLEICH"
...
99 !FI
```

Ist die Bedingung hinter IF erfüllt, so werden die Zeilen zwischen der IF- und der FI-Anweisung ausgeführt, ansonsten wird das Programm sofort hinter der FI-Zeile fortgesetzt.

Daneben existiert selbstverständlich auch die vollständige Form

```
10 !IF X=Y THEN
20:    PRINT "GLEICH'
...
50 !ELSE
60 :   PRINT "UNGLEICH"
...
99 !FI
```

Ist die Bedingung erfüllt, dann wird der Block zwischen IF und ELSE ausgeführt, sonst der Block zwischen ELSE und FI.

Für den Fall, daß mehr als nur zwei Fälle zu unterscheiden sind, bietet Strubs die CASE-Anweisung:

```
10 !CASEOF X<0 THEN
15 :    PRINT "KLEINER ALS 0"
...
40 ! OF X=0 THEN
45 :    PRINT "GLEICH 0"
...
60 ! OF X>0 AND Y <XTHEN
65 : PRINT "X>0 UND Y<X"
...
80 ! ELSE
85 :    PRINT "KEINER DER FÄLLE TRIFFT ZU"
99 ! ECASE
```


Mit dieser Struktur können beliebig viele Fälle unterschieden werden, wobei jedes OF mit einer beliebigen Bedingung verbunden werden kann. Es sollte aber darauf geachtet werden, daß sich die Bedingungen gegenseitig ausschließen (sonst wird das erste Auftreten einer erfüllten Bedingung gewählt). Nach der Bearbeitung des entsprechenden Falles wird das Programm immer hinter ECASE fortgesetzt. Die Möglichkeit, daß keiner der Fälle zutrifft, kann mit Hilfe der ELSE-Anweisung behandelt werden. Ist dies nicht erforderlich, kann der ELSE-Teil auch entfallen.

Damit kommen wir nun zu den Schleifen. Die FOR-Schleife kann wie bisher benutzt werden. Die WHILE-Schleife wird durchlaufen, solange die Bedingung erfüllt ist. Anschließend wird das Programm hinter EWHILE fortgesetzt. Da die Bedingung am Anfang der Schleife abgefragt wird, kann es vorkommen, daß die Schleife auch überhaupt nicht durchlaufen wird. Ein Beispiel:

```
10 ! WHILE X<5 !DO
20 :   PRINT "IMMER NOCH KLEINER ALS 5"
30 : X = X + 1
...
99 !EWHILE
```

```
510 '*****************************
520 '* GESCHACHTELTE LOOP-BLOECKE    *
530 '*****************************
540 '
620 !LOOP 'L1
630 :    PRINT"AEUSSERE LOOP1"
640 :    IF X=1 THEN !EXIT 'LOOP1
650 :    !LOOP 'L2
660 :       PRINT "INNERE LOOP2"
670 :       IF X=0 THEN !EXIT 'LOOP2
680 :    !ELOOP ' L2
690 '    HIER WIRD PROGR. NACH EXIT LOOP2 FORTGESETZT
700 :    X=X+1
710 :    !IF X=2 THEN
720 :       PRINT "LOOP 1 VERLASSEN:":!EXIT 'LOOP1
730 :    !FI
740 :    X=X+1
750 !ELOOP 'L1
760 PRINT"HIER WIRD PROGRAMM NACH EXIT LOOP 1 FORTGESETZT"
770 '
READY.
```

Bild 2. Geschachtelte Loop-Schleife

TRUE

. . .

EXIT-BEDINGUNGEN

. . .

Bild 3. Struktogramm der Loop-Schleife

```
51 PRINT"";"  *****************"
52 PRINT "  * --STRUBS.4 -- *"
55 PRINT "  *    M.TOERK     *"
57 PRINT "  * 4352 HERTEN    *"
58 PRINT "  *****************"
70 IFNOT(PEEK(46)<46OR(PEEK(46)=46ANDPEEK(45)<3))THEN75
73 POKE46,46:POKE45,3:POKE46*256,0:CLR
75 :
80 EA=46*256+1

READY.

8860 PRINT""
8870 PRINT "**********************"
8880 PRINT "** ZURUECK MIT:     **"
8882 PRINT "** ' ! ' [RETURN]   **"
8940 PRINT "**********************"

READY.
```

Dies Änderung sind für die Anpassung von Strubs an den VC 20 (mit mindestens 16 KByte Erweiterung) erforderlich.

```
STRUBS.4 OBJEKTPROGRAMM:

5 REMSTRUBS4/4.9.83
51 PRINT"J";TAB(10);"******************"
52 PRINTTAB(10);"* --STRUBS.4 -- *"
55 PRINTTAB(10);"*    M.TOERK     *"
57 PRINTTAB(10);"* 4352 HERTEN    *"
58 PRINTTAB(10);"******************"
70 IFNOT(PEEK(46)<40OR(PEEK(46)=40ANDPEEK(45)<3))THEN75
73 POKE46,40:POKE45,3:POKE40*256,0:CLR
75 :
80 EA=40*256+1
100 GOSUB45060
140 GOTO40050
250 IFPEEK(NC)=BLTHENNC=NC+1:GOTO250
260 C=PEEK(NC)
265 IFC<>KOTHEN320
280 NC=NC+1:C=PEEK(NC):IFCANDC<>KOTHEN280
290 IFCTHENNC=NC+1:C=PEEK(NC)
295 IFC=BLTHEN250
320 IFC<>TETHENNC=NC+1:RETURN
350 Z$=Z$+CHR$(C):NC=NC+1:C=PEEK(NC):IFCANDC<>TETHEN350
370 NC=NC+1
390 RETURN
550 IFLEN(Z$)<4THENRETURN
555 PRINTFNAD(ZA+2)
560 AA=AA+LEN(Z$)+2
565 H%=AA/256
570 PRINT#1,CHR$(AA-256*H%);CHR$(H%);Z$;
580 RETURN
750 T$=""
790 :
795 C=PEEK(NC):IFC=DPORC=KMORC=BLORC=0THEN811
800 NC=NC+1:T$=T$+CHR$(C)
810 GOTO790
811 :
820 NC=NC+1:IFC=BLTHENGOSUB250
830 RETURN
1050 GOSUB750
1120 IFNOT(T$="THIS")THEN1131
1125 H=FNAD(ZA+2)
1130 GOTO1175
1131 :
1140 FORI=0TOMP:IFMA$(I)<>T$THENNEXT
1160 IFI>MPTHENER=2:GOTO8050:
1170 H=MA%(I)+DI
1175 :
1180 Z$=Z$+MID$(STR$(H),2)
1190 RETURN
1550 GOSUB750
1560 FORI=0TOBM:IFT$<>BE$(I)THENNEXT
1565 IFI>BMTHENER=0:GOTO8050
1567 B$=BE$(I):IFI=3THENB$="IF"
1569 I=I+1
1570 ONIGOSUB1600,1680,1640,2010,2040,2100,2160,2210,2260,
2400,1710,1740,1810,1860
1574 PRINTFNAD(ZA+2);
1575 IFIN=0THENPRINTTAB(TA);B$:RETURN
1577 IFIN=1THENPRINTTAB(TA);B$:TA=TA+1:RETURN
1579 IFIN=2THENPRINTTAB(TA-1);B$:RETURN
1581 IFIN=3THENTA=TA-1:PRINTTAB(TA);B$:RETURN
1586 RETURN
1600 IFSP>SMTHENER=3:GOTO50000
1605 IFLP>LMTHENER=5:GOTO50000
1610 S%(SP)=LP:SP=SP+1:LO%(LP,0)=FNAD(ZA+2)-DI:LP=LP+1
1615 IN=1:RETURN
1640 SP=SP-1:IFSP<0THENER=1:GOTO50000
1650 LO%(S%(SP),1)=FNAD(ZA+2)-DI
1660 IN=3:RETURN
1680 IN=0:RETURN
```


Listing. Das Objektprogramm Strubs

Von der WHILE-Schleife unterscheidet sich die REPEAT-Schleife in zwei Punkten: Erstens wird die Schleife durchlaufen, bis die Bedingung erfüllt ist, also solange sie nicht erfüllt ist. Zweitens wird die Bedingung erst am Ende der Schleife abgefragt, so daß die Schleife immer mindestens einmal durchlaufen wird. In diesem wie im nächsten Beispiel bezieht sich die Zeile 30 auf den Fall, daß X beim Eintritt in die Schleife größer als 5 ist:

```
10 ! REPEAT
20 :   PRINT "X KLEINER ALS 5"
30 : PRINT "VIELLEICHT ABER
AUCH NICHT"
40 : X = X+1
...
99 ! UNTIL X >= 5
```

Eine weniger weit verbreitete, aber sehr mächtige Schleifenstruktur stellt die LOOP-Schleife dar (sie befindet sich zum Beispiel in der Programmiersprache ADA):

```
10 ! LOOP
30 :   PRINT "EVENTUELL GROES-
SER ALS 5"
40 :   IF X>=5 THEN !EXIT
50 :   PRINT "KLEINER ALS 5"
60 :   X = X+1
...
99 !ELOOP
```

Verlassen einer Endlosschleife

Es handelt sich dabei um eine Endlosschleife, welche mit Hilfe des Befehls EXIT verlassen werden kann. Diese Schleife bietet im wesentlichen zwei Vorteile: Zum einen muß die Bedingung nicht entweder am Anfang oder am Ende der Schleife stehen, sondern kann an jeder beliebigen Stelle innerhalb des Blockes abgefragt werden. Darüber hinaus ist das Beenden der Schleife nicht nur von einer Bedingung abhängig, sondern die LOOP-Schleife kann beliebig viele EXIT-Anweisungen enthalten (dadurch wird nicht die oben erwähnte Forderung nach nur einem Ausgang verletzt, da das Programm in allen Fällen hinter dem ELOOP fortgesetzt wird). Damit eignet sich diese Konstruktion insbesondere gut für die Behandlung von Ausnahmen wie zum Beispiel von Eingabebefehlen etc. (eine Angelegenheit, die zum Beispiel in Pascal recht umständlich sein kann, falls man auf GOTOs verzichten will oder muß).

In Bild 2 (das Zeichen ' kennzeichnet Kommentare) sehen Sie ein Beispiel für geschachtelte LOOP-

Schleifen. Die Ausführung einer EXIT-Anweisung bewirkt die Fortsetzung des Programms bei der ersten Zeile hinter derjenigen Schleife, welche diese EXIT-Anweisung am nächsten umschließt. Im Beispiel enthält die äußere Schleife zwei EXIT-Anweisungen — eine davon vor, die andere hinter der inneren Schleife. Die innere Schleife enthält eine EXIT-Anweisung. Grafisch lassen sich blockstrukturierte Programme am besten durch Struktogramme — anstelle der verbreiteten Flußdiagramme — darstellen. Das Struktogramm für die LOOP-Schleifen finden Sie in Bild 3. Über die Diagramme der anderen Strukturen und den Umgang mit Struktogrammen können Sie sich an anderer Stelle in dieser Zeitschrift oder in den unten aufgeführten Büchern informieren. Kommen wir nun zu den Modulen. Dabei handelt es sich um besondere Blöcke, die ein bestimmtes Teilproblem — beispielsweise das Zeichnen einer Linie in einem Grafikprogramm — unter möglichst weitgehender Unabhängigkeit vom restlichen Programmtext bearbeiten. Stellen Sie sich vor, Sie finden in einer Zeitschrift ein Pascal-Programm zur Einstellung von Grafiken. Dieses Programm benutzt zum Beispiel die Anweisung PLOT (X,Y) zum Zeichnen eines Punktes mit den Koordinaten X und Y. Ihr Freund möge eine Sprache Super-Pascal besitzen, die diese Anweisung standardmäßig enthält. Er tippt das Programm ein, es läuft — fertig. Sie selbst besitzen aber nur ein mageres Mini-Pascal, das diesen Befehl nicht kennt. Nun, mit Pascal ist das kein Problem: Sie schreiben sich eine Procedur PLOT (X,Y) fügen diese in das Programm ein — fertig. An dem Programmtext selbst brauchen Sie nicht die geringste Änderung vorzunehmen. Ja, brauchen ihn nicht einmal näher anzusehen. Woran liegt das?

Vom Problem her — dem Erstellen einer Grafik — ist das Zeichnen eines Punktes das Zeichnen eines Punktes. Das einzige, was interessiert, ist, daß dazu zwei Koordinaten erforderlich sind. Dieser Tatsache trägt die Sprache Pascal dadurch Rechnung, daß sie keinen Unterschied macht zwischen dem Aufruf von vorgegebenen Standardanweisungen und selbst definierten Prozeduren.

Wenn Sie in einem Basic-Programm irgendwo eine Zeile PRINT "TEXT" stehen haben, erwarten Sie selbstverständlich, daß

```
1710 GOSUB1600:RETURN
1740 GOSUB1640:RETURN
1810 GOSUB1600:RETURN
1860 GOSUB1640:RETURN
2010 IFSP>SMTHENER=3:GOTO50000
2011 IFIP>IMTHENER=4:GOTO50000
2020 S%(SP)=IP:IP=IP+1:SP=SP+1
2025 IN=1:RETURN
2040 IFSP<1THENER=1:GOTO50000
2041 IFIP>IMTHENER=4:GOTO50000
2044 I%(S%(SP-1))=FNAD(ZA+2)+1-DI
2050 S%(SP-1)=IP:IP=IP+1
2052 IN=2:RETURN
2100 IFSP<1THENER=1:GOTO50000
2105 SP=SP-1:I%(S%(SP))=FNAD(ZA+2)-DI
2107 IN=3:RETURN
2160 IFSP>SMTHENER=3:GOTO50000
2165 S%(SP)=-1:SP=SP+1
2170 GOSUB2010
2180 IN=1:RETURN
2210 GOSUB2040
2230 GOSUB2010
2240 IN=2:RETURN
2260 H=FNAD(ZA+2)-DI
2270 :
2275 IFSP<1THENER=1:GOTO50000
2280 SP=SP-1:I=S%(SP)
2290 IFI<0THEN2311
2300 I%(I)=H
2310 GOTO2270
2311 :
2320 IN=3:RETURN
2400 :
2410 IFMP>MMTHENER=6:GOTO50000
2415 IFCANDC<>LATHENGOSUB250:GOTO2415
2420 IFCTHENGOSUB750
2423 IFCTHENGOSUB250
2425 IFC<48ORC>57THENER=9:GOTO8050
2430 MA$(MP)=T$:H=C
2440 GOSUB750
2450 MA%(MP)=VAL(CHR$(H)+T$)-DI
2460 MP=MP+1
2470 IFC=0THEN2481
2480 GOTO2400
2481 :
2485 IN=0:RETURN
2550 GOSUB750
2560 FORI=0TOBM:IFT$<>BE$(I)THENNEXT
2565 IFI>BMTHENER=0:GOTO8050
2568 I=I+1
2570 ONIGOSUB2590,2685,2630,3010,3090,3190,3260,3310,3360,
3400,3450,3550,3580,3600
2575 RETURN
2590 IFC=0THENZ$=Z$+":"
2595 S%(SP)=LP:SP=SP+1:LP=LP+1
2597 RETURN
2630 SP=SP-1
2640 Z$=Z$+GT$+MID$(STR$(LO%(S%(SP),0)+DI),2)+NU$
2642 GOSUB550
2647 L=PEEK(ZA+2)+1:H=PEEK(ZA+3):IFL>255THENL=0:H=H+1
2648 Z$=CHR$(L)+CHR$(H)+":"
2650 RETURN
2685 B$="":IFRIGHT$(Z$,1)<>CHR$(167)THENB$=GT$
2693 Z$=Z$+B$+MID$(STR$(LO%(S%(SP-1),1)+DI+1),2)
2695 RETURN
3010 Z$=Z$+IC$+NO$+"("+CHR$(C)
3020 GOSUB250:IFC<>THANDCTHENZ$=Z$+CHR$(C):GOTO3020
3030 Z$=Z$+")"+CHR$(TH)+MID$(STR$(I%(IP)+DI),2)
3036 IP=IP+1:C=0:RETURN
3090 Z$=Z$+GT$+MID$(STR$(I%(IP)+DI),2)+NU$
```

Listing. Das Objektprogramm Strubs (Fortsetzung)

```
3100 GOSUB550
3120 L=PEEK(ZA+2)+1:H=PEEK(ZA+3):IFL>255THENL=0:H=H+1
3130 Z$=CHR$(L)+CHR$(H)+":"
3140 IP=IP+1:RETURN
3190 L=PEEK(ZA+2):H=PEEK(ZA+3)
3200 Z$=CHR$(L)+CHR$(H)+":"
3210 RETURN
3260 GOSUB3010:RETURN
3310 GOSUB3090
3320 Z$=LEFT$(Z$,LEN(Z$)-1)
3330 GOSUB3010
3340 RETURN
3360 GOSUB3190
3370 RETURN
3400 Z$="":C=0:RETURN
3450 GOSUB2590
3460 Z$=Z$+IC$+NO$+"("
3470 IFC<>BEANDCTHENZ$=Z$+CHR$(C):GOSUB250:GOTO3470
3480 Z$=Z$+")"+CHR$(TH)
3490 Z$=Z$+MID$(STR$(LO%(S%(SP-1),1)+DI+1),2)
3495 C=0:RETURN
3550 GOSUB2630:RETURN
3580 GOSUB2590:RETURN
3600 Z$=Z$+IC$+NO$+"("
3610 IFCTHENZ$=Z$+CHR$(C):GOSUB250:GOTO3610
3620 SP=SP-1:IN=3
3630 Z$=Z$+")"+CHR$(TH)+MID$(STR$(LO%(S%(SP),0)+DI),2)
3640 RETURN
4060 Z$=CHR$(PEEK(ZA+2))+CHR$(PEEK(ZA+3))
4080 NC=ZA+4:GOSUB250
4090 IFC=DPTHENGOSUB250
4100 IFNOT(C=LA)THEN4110
4105 GOSUB750:IFC=DPTHENGOSUB250
4108 IFC=0THENZ$=Z$+":"
4110 :
4115 NC=NC-1:IFC=0THENZ$=Z$+NU$
4130 :IFC=0THEN4397
4132 GOSUB250
4150 IFNOT(C=BE)THEN4359
4155 GOSUB2550
4358 GOTO4378
4359 :
4360 IFC=LATHENGOSUB1050
4378 :
4380 Z$=Z$+CHR$(C)
4396 GOTO4130
4397 :
4398 RETURN
5050 PRINT"{CLR}   ***** UEBERSETZEN    *****{DOWN}{DOWN}"
5052 IFNOT(FNAD(EA)<EA+50ORFNAD(EA)>EA+83)THEN5054
5053 PRINT"KEIN PROGRAMM VORHANDEN":GOSUB49550:RETURN
5054 :
5058 PRINT"BITTE DISK EINLEGEN {DOWN}{DOWN} "
5060 PRINT"NAME FUER OBJEKT-PROGRAMM"
5065 POKE198,1:POKE631,34
5070 INPUTF$
5080 OPEN1,8,1,F$+",P,W":OPEN15,8,15
5090 INPUT#15,E,E$:IFE=0THEN5101
5095 PRINT"DISK ERR:";E;E$
5096 INPUT"NEUER VERSUCH";Z$
5098 CLOSE1:CLOSE15
5099 IFZ$<>"J"THENRETURN
5100 GOTO5060
5101 :
5120 AA=EA
5130 PRINT#1,CHR$(AAAND256);CHR$(AA/256);
5135 PRINT"1.LAUF"
5136 TA=7
5140 GOSUB5555
5143 IFSP>0THENPRINTSP;:ER=8:GOTO50000
```

Listing.
Das Objektprogramm Strubs (Fortsetzung)

dadurch nicht 50 Zeilen weiter der Wert der Variablen A verändert wird. Entsprechend sorgt nun Pascal dafür, daß eine selbst definierte Prozedur genausowenig Auswirkungen auf andere Programmteile hat wie der Aufruf einer Standard-Anweisung. Die interne Arbeitsweise einer solchen Prozedur wird vor der Programmumgebung genauso versteckt, wie dies bei der internen Arbeitsweise von im Sprachumfang enthaltenen Anweisungen der Fall ist. Entsprechend nennt man dieses Konzept auch »Information Hiding«. Programmiersprachen wie ADA, MODULA oder SIMULA bieten in dieser Hinsicht noch sehr viel weitergehende Möglichkeiten als Pascal.

Schnittstellen:

Der Datenaustausch mit der Umgebung eines Moduls erfolgt über genau definierte Schnittstellen. Bei einer solchen Schnittstelle handelt es sich um eine Menge derjenigen Annahmen, die die Programmumgebung über ein Modul macht — das heißt welche Daten es als Eingabe erwartet, welche Daten es daraufhin wieder ausgibt und welche anderen Module es seinerseits benötigt.

Modulbibliothek:

Die relative Eigenständigkeit solcher Module sorgt nun nicht nur für einfache Änderbarkeit und Erweiterbarkeit, sondern ermöglicht auch das Anlegen einer sogenannten Modulbibliothek. Eine solche Bibliothek enthält eine Reihe von Programmbausteinen, die je nach Bedarf in zu entwickelnde Programme eingefügt werden können. Dabei kann es sich um Sortierroutinen, Grafik-Routinen, mathematische und statistische Routinen und so weiter handeln. Aber auch die Entwicklung von Spielen läßt sich auf diese Weise vereinfachen: Man kann Bibliotheken fertiger Sprites, von eigenen Zeichensätzen oder von diversen Soundroutinen anlegen.

Das wichtigste Hilfsmittel zur Unterstützung modularer Programmentwicklung stellen sicherlich die lokalen Variablen dar. Leider gibt es solche nicht in Basic und auch Strubs kann keine lokalen Variablen bieten. So ist es auch weiterhin erforderlich, beim Einsetzen oder Ändern eines Moduls darauf zu achten, ob und an welchen Stellen Variablen des Moduls in anderen Programmteilen benutzt werden, und gegebenenfalls Umbenennungen vorzunehmen. Der zweite große

Nachteil von Basic — die leidigen Zeilennummern — braucht uns dagegen nur noch wenig zu beschäftigen. Strubs bietet alle Möglichkeiten, die erforderlich sind, um ein Programm vollkommen unabhängig von Zeilennummern zu schreiben. Als erstes sind da natürlich die oben besprochenen Kontrollstrukturen zu nennen. Darüber hinaus können bei allen Sprüngen Zeilennummern durch Labels (Marken) ersetzt werden. Solche Labels werden durch das Zeichen »£« gekennzeichnet und abgeschlossen durch ein Leerzeichen, Doppelpunkt, Komma oder Zeilenende. Die dürfen zwar reservierte Basic-Worte enthalten, dann können sich aber wegen der in der letzten Folge erwähnten Tokens bei der Ausgabe der Markentabelle seltsame Effekte ergeben. Die Labels werden definiert, indem sie an den Anfang einer Zeile gesetzt werden und können beliebig lang sein:

```
10  AUSGEBEN:
20 : PRINT "X:";X
30  RETURN
...
200 X=1:GOSUB £X-AUSGEBEN
210 X=2:GOSUB £X-AUSGEBEN
```

Schließlich bietet Strubs noch die Möglichkeit relativer Sprünge. Diese dienen vor allem dazu, kurze Schleifen innerhalb einer einzigen Zeile zu konstruieren, ohne dafür extra ein Label zu definieren:

```
90 NC=NC+1:C=PEEK(NC):IF C>0 THEN Z$+CHR$(C): GOTO £THIS
```

Der Befehl GOTO £THIS bewirkt einen Sprung an den Anfang derjenigen Zeile, in der dieser Befehl steht.

Da bei der Arbeit mit Strubs Quellprogramme in der Regel weit umfangreicher als die Objektprogramme sind, bietet Strubs die EXTERN-DEKLARATION, die es ermöglicht, Module und Programmteile getrennt zu übersetzen und erst auf der Objektprogrammebene zusammenzufügen. Hierbei müssen die einzelnen Programmteile allerdings verschiedene Zeilennummern belegen. In der Extern-Deklaration wird ein Name vereinbart, unter dem ein Programm ein externes Modul ansprechen kann. Diesen Namen wird die Einsprungadresse (bei Maschinenprogrammen) beziehungsweise die Zeilennummer bei Basic-Routinen zugewiesen:

```
20 REM VEREINBARUNG:
30 ! EXT: £MAPRO:740,£PLOT:50000
```

Fortsetzung auf Seite 126

```
5145 PRINT"2.LAUF"
5150 GOSUB6550
5160 PRINT#1,CHR$(0);CHR$(0);
5180 CLOSE1:PRINT"{DOWN}**";EP;" ERRORS **":GOSUB49550
5190 RETURN
5555 ZA=EA
5570 IFNOT(ZA<>0)THEN5931
5580 NC=ZA+4:C=PEEK(NC):NC=NC+1
5585 IFC=DPTHENGOSUB250
5590 IFC=LATHENGOSUB6050:IFC=DPTHENGOSUB250
5620 IFC=BETHENGOSUB1550
5920 ZA=FNAD(ZA)
5930 GOTO5570
5931 :
5935 RETURN
6050 IFMP>MMTHENER=6:GOTO50000
6070 GOSUB750
6100 MA$(MP)=T$:MA%(MP)=FNAD(ZA+2)-DI:MP=MP+1
6120 RETURN
6550 ZA=EA:Z1=FNAD(ZA)
6560 LP=0:SP=0:IP=0
6580 :
6585 IFNOT(PEEK(ZA+4)<>KO)THEN6650
6590 GOSUB4060
6600 GOSUB550
6650 :
6655 ZA=Z1:Z1=FNAD(Z1)
6660 IFNOT(Z1=0)THEN6580
6680 RETURN
8050 PRINT"ERROR IN";FNAD(ZA+2),ER$(ER)
8060 IFEP<EMTHENER%(EP,0)=FNAD(ZA+2)-DI:ER%(EP,1)=ER:EP=EP+1
8080 Z$=LEFT$(Z$,2)+"***** ERR:"+ER$(ER)+"********"
8090 C$=NU$:C=0
8099 RETURN
8860 PRINT"{CLR}{DOWN}{DOWN}{DOWN}{DOWN}{DOWN}"
8870 PRINTTAB(9);"**********************"
8880 PRINTTAB(9);"** ZURUECK MIT:    **"
8882 PRINTTAB(9);"** ' ! ' [RETURN]  **"
8940 PRINTTAB(9);"**********************"
8950 POKE44,EA/256:POKEEA-1,0:CLR:END
8990 END
40050 PRINT"{CLR}";TAB(10);"*****************"
40052 PRINTTAB(10);"* -- STRUBS  -- *"
40053 PRINTTAB(10);"*  PRECOMPILER  *"
40055 PRINTTAB(10);"* BITTE WAEHLEN *"
40058 PRINTTAB(10);"*****************"
40060 PRINT"{DOWN}{DOWN}{DOWN}{RVS}E{OFF}DIT"
40070 PRINT"{DOWN}{DOWN}{RVS}U{OFF}EBERSETZEN"
40080 PRINT"{DOWN}{DOWN}{RVS}M{OFF}ARKEN-TABELLE AUSGEBEN"
40090 PRINT"{DOWN}{DOWN}{RVS}F{OFF}EHLER-TABELLE AUSGEBEN"
40100 PRINT"{DOWN}{DOWN}{RVS}S{OFF}CHLUSS"
40160 GETZ$:IFZ$=""THEN40160
40170 IFZ$="E"THEN8860
40180 IFZ$="U"THENGOSUB5050:GOTO40050
40190 IFZ$="S"THENEND
40195 IFZ$="M"THENGOSUB48050:GOTO40050
40200 IFZ$="F"THENGOSUB49050:GOTO40050
40495 GOTO40050
45060 MM=99:DIMMA$(MM),MA%(MM):MP=0
45135 LM=140:DIMLO%(LM,1):LP=0
45145 IM=270:DIMI%(IM):IP=0
45190 SM=60:DIMS%(SM):SP=0
45220 DI=32766
45250 DP=ASC(":"):KO=ASC("'"):LA=ASC("£"):NU$=CHR$(0):BL=ASC(" ")
45253 BE=ASC("!"):TE=34:GT$=CHR$(137)
45254 IC$=CHR$(139):TH=167:NO$=CHR$(168):KM=44
45265 BM=13:DIMBE$(BM)
45270 FORI=0TOBM:READBE$(I):NEXT
45271 BE$(3)=IC$
45272 DATALOOP,EXIT,ELOOP,IF,ELSE,FI
```

Listing. Das Objektprogramm Strubs (Fortsetzung)

```
45273 DATACASEOF,OF,ECASE,EXT
45274 DATAWHILE,EWHILE,REPEAT,UNTIL
45410 DEFFNAD(X)=PEEK(X)+256*PEEK(X+1)
45480 EM=40:DIMER%(EM,1):EP=0:DIMER$(40)
45500 FORI=0TO9:READER$(I):NEXT
45510 DATA"FALSCHER BEFEHL","BLOCKSCHACHTELUNG: ANFANG FEHLT"
45511 DATA"UNDEFINIERTE MARKE","STACK VOLL"
45512 DATA"ZU VIELE IF/ELSE/CASE/OF","ZU VIELE LOOP/WHILE/REPEAT"
45513 DATA"ZU VIELE MARKEN",,"BLOCK NICHT GESCHLOSSEN"
45514 DATA"EXTERN DECLARATION"
45600 I=0:READW
45610 POKE704+I,W:I=I+1:READW:IFW<256THEN45610
45620 DATA32,115,0,8,201,33,240,4,40,76,231,167
45630 DATA169,8,133,44,169,138,76,231,167,999
45650 FORI=0TO10:READW:POKE750+I,W
45660 NEXT
45670 SYS750
45680 DATA169,192,141,8,3,169,2,141,9,3,96
45999 RETURN
48050 IFMP=0THENRETURN
48055 H=0
48057 PRINT"▯▯          ** MARKENTABELLE AUSGEBEN **"
48060 INPUT"▯▯ AUF DRUCKER (J/N)";B$
48070 IFNOT(B$="J")THEN48091
48075 PRINT" DRUCKER AN?":GOSUB49550
48080 OPEN1,4
48090 GOTO48104
48091 :
48100 OPEN1,3
48102 H=-1
48104 :
48120 FORI=0TOMP-1
48140 PRINT#1,MA%(I)+DI,MA$(I)
48150 IFI-INT(I/10)*10=0THENIFIANDHTHENGOSUB49550
48180 NEXT
48185 CLOSE1:GOSUB49550
48190 RETURN
49050 IFEP=0THENRETURN
49055 H=0
49057 PRINT"▯▯          ** FEHLERTABELLE AUSGEBEN **"
49060 INPUT"▯▯ AUF DRUCKER (J/N)";B$
49070 IFNOT(B$="J")THEN49091
49075 PRINT"▯ DRUCKER AN?▯▯":GOSUB49550
49080 OPEN1,4
49090 GOTO49104
49091 :
49100 OPEN1,3
49102 H=-1
49104 :
49110 PRINT#1,EP;" ERRORS"
49120 FORI=0TOEP-1
49140 PRINT#1,ER%(I,0)+DI;ER$(ER%(I,1))
49150 IFI-INT(I/10)*10=0THENIFIANDHTHENGOSUB49550
49180 NEXT
49185 CLOSE1
49191 GOSUB49550
49190 RETURN
49550 PRINT"->▯▯▯";
49560 GETB$:IFB$=""THEN49560
49570 RETURN
50000 PRINT"▯* FEHLER BEHEBEN, DANN NEU VERSUCHEN *"
50008 PRINT:PRINTER$(ER);" IN ";FNAD(ZA+2)
50010 PRINT#1,CHR$(0);CHR$(0);
50020 CLOSE1
50030 GOSUB49550
50040 GOSUB49050
50050 RUN

READY.
```

Listing. Das Objektprogramm Strubs (Schluß)

Fortsetzung von Seite 121

...
90 REM AUFRUF:
99 SYS £MAPRO: X=13:Y=90:GOSUB £PLOT

Kommen wir abschließend zur Dokumentation: Vom Hobby-Programmierer kann kein Mensch erwarten, daß er Berge von Dokumentationsmaterial anlegt, die den Umfang des Programmtextes um ein Vielfaches übersteigen. Deshalb ist es gerade hier wichtig, Programme weitgehend selbstdokumentierend zu schreiben. Im Gegensatz zu höheren Programmiersprachen mit ihren zahlreichen Deklarationspflichten ist der Basic-Programmierer nahezu ausschließlich auf Kommentare angewiesen. Da Strubs Kommentare bei der Übersetzung eleminiert, stehlen diese weder Speicherplatz noch Laufzeit. Der Programmierer kann also ohne Bedenken einen exzessiven Gebrauch von Kommentaren machen.

Kommentare werden gekennzeichnet durch das Zeichen » ´ «. Steht dieses Zeichen direkt am Zeilenanfang, so wird die ganze Zeile gelöscht. Sonst wird der Programmtext bis zum zweiten » ´ « oder bis zum Zeilenende überlesen. Außer innerhalb von Befehls- und Markennamen können Kommentare an jeder beliebigen Programmstelle eingefügt werden. Kommentare, die in das Objektprogramm übernommen werden sollen, können wie bisher mit REM in den Programmtext eingefügt werden. Beispiel:

10 ´DIESE ZEILE WIRD VOLLSTÄNDIG GELÖSCHT
20 A´US´G´ABE´$="ENTSPRICHT AG$" ´KOMMENTAR

Die Lesbarkeit von strukturierten Programmen wird verbessert durch das Einrücken von Zeilen entsprechend der Blockstruktur. Hierzu dient der Tabulator (Bild 2): Ein Doppelpunkt am Zeilenanfang gefolgt von Leerzeichen. Für die Ungeduldigen ist das Opjektprogramm von Strubs bereits abgedruckt (Listing). In der nächsten Ausgabe werden wir auf die praktische Programmentwicklung mit Hilfe von Strubs eingehen.

(Matthias Törk)

Literatur

* N. Wirth: Systematisches Programmieren, Teubner, Stuttgart 1978
* Kimm, R./Koch,W./Simonsmeier,W./Tontsch,F.: Einführung in Software Engineering, De Gruyter, Berlin, New York 1979
* Schnupp, P./FLoyd, C.: Software: Programmentwicklung und Projektorganisation, De Gruyter, Berlin, New York 1976
* Nagl, M.: Einführung in die Programmiersprache ADA, Vieweg, Braunschweig, Wiesbaden 1982

Welche Hausnummer hat der Kölner Dom?

Meist sind sie in der wärmeren Jahreszeit unterwegs, um die Gegend zu »erkunden« — die Orientierungsfahrer. Wer schon einmal an dieser Art Autorallye teilgenommen hat, weiß, wieviel Spaß es macht, eine — häufig verschlüsselt — vorgegebene Strecke abzufahren, unterwegs Aufgaben zu lösen und dabei die eigene Originalität und Kreativität gründlich unter Beweis zu stellen.

»Eingeborene« haben einen enormen Heimvorteil

Wie lästig ist es aber zum Schluß, nachdem auch die letzten von ihren Irrfahrten eingetrudelt sind, die Auswertung abzuwarten. Das merkte auch Ingo Molitor als er an einer Orientierungsfahrt im Kölner Raum teilnahm. Nach 240 km Fahrtstrecke und enthusiastischem Eifer, alle unterwegs gestellten Aufgaben herauszubekommen, wollte er am liebsten auf der Stelle wissen, wie er im Vergleich zu den anderen Teilnehmern abgeschnitten hatte. Vielleicht war er sogar Erster geworden?

Gezwungenermaßen mußte er seine Ungeduld zügeln — zwei Stunden lang. Und das Ergebnis verbesserte auch nicht gerade seine mittlerweile auf den Nullpunkt gesunkene Laune: Platz 12. Doch kurz vor der Siegerehrung wußte man es besser. Nachdem alle bis dahin verkündeten Ergebnisse von Berechnungs- und Auswertungsfehlern bereinigt waren, besetzte er Platz 8 — nur richtig freuen konnte sich Ingo Molitor jetzt nicht mehr über seinen unfreiwilligen Aufstieg. Ihn beschäftigte schon längst ein ganz anderer Gedanke: So etwas durfte nie wieder vorkommen. Wozu hatte er denn einen Computer — der müßte doch geeignet sein, die bei einer Orientierungsfahrt übliche Spontanität und Freude auch bis zuletzt aufrechtzuerhalten.

Bereits am nächsten Tag begab sich der »Rallyeverbesserer« ans Werk, er schrieb das notwendige Programm auf seinem VC 20. Wenige Tage später »stand« es. Nur jetzt ließ die nächste Orientierungsfahrt auf sich warten. Der Termin war leider nicht per Computer zu steuern.

Sommer und Herbst gingen ins Land — und aufregender als das bevorstehende Weihnachtsfest war der 18. Dezember für Ingo Molitor, denn heute sollte eine Rallye stattfinden. Namen der Fahrer, Beifahrer, Autotypen, Startnummern und so weiter hatte er bereits gespeichert. Mit einem Ausdruck der Startliste in der Hand stand er voller Lampenfieber auf dem Parkplatz, wo die »Jungfernfahrt« seines Programms beginnen sollte. Das Lampenfieber war nicht ganz unbegründet, denn in einem »Anfall« totaler Sicherheit hatte man sich vollkommen auf den Computer verlassen — es gab keinerlei handschriftliche Aufzeichnungen oder Notizen.

Doch die Bedenken gingen zunächst unter, denn das Wesentliche war gegeben: alle Teilnehmer hatten einen Riesenspaß, dafür sorgten schon die gestellten Aufgaben. So mußte man beispielsweise erkunden »Welche Hausnummer hat der Kölner Dom? Wieviele Stufen führen zum Turm?« Darüber hinaus sollte sich jeder eine Quittung darüber besorgen, daß er in einem fremden Haus ein Fenster geputzt hatte. Und dann ging's unter der Erde weiter, es galt herauszufinden, wie die Patenstadt eines bestimmten Wagens der Linie 16 heißt.

Sieger bei der Ein-Mann-Rallye

Da Ingo Molitor zu den Organisatoren gehörte, durfte er leider nicht mitmachen — aber eine Rallye sollte er auch noch erleben. Mit einigen Freunden fuhr er zum Zielpunkt, um bei einer gemütlichen Tasse Kaffee den nächsten Coup vorzubereiten: Jetzt stand Eierkochen auf dem Plan. Gegen 15 Uhr fuhr Ingo Molitor vom Zielpunkt zu einem der unterwegs festgelegten Kontrollpunkte und drückte jedem Team ein unverwechselbar markiertes rohes Ei »in die Hand«. Es sollte bis zum Ziel auf irgendeine Weise in ein gekochtes verwandelt werden — Kreativität war Trumpf.

Nun blieb für den Organisator nicht mehr viel zu tun. Am Zielpunkt mußte der VC 20 mit Programm und Drucker für das Eintreffen der Teilnehmer vorbereitet werden. Vor 18 Uhr konnten auch die Schnellsten nicht da sein; Ingo Molitor hatte noch viel Zeit. Doch das änderte sich schlagartig: Als er nämlich das Hauptprogramm starten wollte, erschien auf dem Bildschirm nur eine sehr klare aber in dem Moment außerordentlich freche Meldung »LOAD ERROR«. Trotz zwölf Versuchen und gutem Zureden, der Computer blieb bei seiner Ansicht. Keine Daten — keine Auswertung ..., das wäre die logische Folge gewesen.

Dies war der Startschuß für Ingo Molitors Privat-Rallye. »Streng« nach den Vorschriften der Straßenverkehrsordnung fuhr er nach Hause, um eine Sicherungskopie des Programms zu holen. Er blieb auch Gewinnen dieser nicht angemeldeten Ein-Mann-Rallye.

Die Auswertung klappte dann bilderbuchartig: Keine mühselige Rechnerei für fünf Leute, die damit systematisch 20 andere entsetzlich langweilen. Eher ein Spiel für einen, an dem sich alle beteiligten, denn die nach dem Eintreffen eines jeden Teams ausgedruckten Zwischenlisten heizten die Stimmung ganz schön an.

Die Lösung der Titelfrage

Der Kölner Dom!
Wo steht die Hausnummer?

Übrigens: Die rohen Eier waren fast alle gekocht. Ein Teilnehmer hatte es jedoch — mangels anderer Gelegenheit vorgezogen, zwischendurch ab und zu anzuhalten, um es auf den heißen Motor zu legen. Auch so wird ein Ei hart ohne daß es gekocht ist! Ein anderer Teilnehmer hat einen Geschäftsmann recht beglückt, indem er dessen Schaufensterscheibe gegen Quittung putzte.

(Ingo Molitor/kg)

Programmbeschreibung

Das Programm für die Auswertung einer Orientierungsfahrt läuft auf einem VC 20 mit 32 KByte RAM, Datasette und einem Epson-RX80-Drucker mit VC 20 Interface.

Es besteht aus zwei Teilen und unterstützt die Vorbereitung und Auswertung einer Autorallye beziehungsweise Orientierungsfahrt mit bis zu 100 Teilnehmern. Der erste Programmteil steuert die Datenerfassung und Speicherung, der zweite Teil beinhaltet die Auswertung. Je nach Speicherausbau können auch die Daten von mehr als 100 Teams nach Änderung der DIM-Zeilen verarbeitet werden.

1. Programmteil

Der erste Teil dient zur Erfassung der Teams und der Fahrzeugtypen. Nach Eingabe des Tagesdatums meldet sich der VC 20 mit einem Menü und die Eingabe kann beginnen.

1. Eingabe Zeilen 100 bis 300

Anfangs wird nach dem Team gefragt, also nach dem Fahrer und Beifahrer. Bei längeren Namen sollten die Vornamen abgekürzt werden. Nach der Abfrage des Fahrzeugtyps muß zuletzt zur Überwachung der Startgeld-Zahlung »J« für »bezahlt« und »N« für »nicht bezahlt« eingegeben werden. Gibt man zu allen Punkten nur »Return« ein, so verläßt das Programm in Zeile 260 den Eingabeteil und kehrt zum Menü zurück.

2. Druckliste Zeilen 350 bis 880

In Zeile 360 ist ein Wartepunkt gesetzt, der das Programm solange stoppt, bis der Drucker angeschaltet wurde, um eine eventuelle Fehlermeldung zu vermeiden. Zeile 380 Open 4,4,0 simuliert auf dem Epson-RX 80 den Commodore VC-1525. Die speziellen Möglichkeiten des

```
5 POKE36879,25:POKE650,128
7 :
10 :
20 REM AUTORALLEY BY INGO MOLITOR
30 PRINT"TAGESDATUM":INPUTDA$
40 REM BIS ZU 100 TEAM'S
50 :
60 :
70 DIMTE$(100),FT$(100),BE$(100)
80 GOTO1500:REM ANFANG DES PROGRAMMS
90 :
100 REM EINGABETEIL
110 :
120 :
130 X=1
140 :
150 :
160 PRINT"EINGABETEIL"
170 PRINT"──────────"
180 :
190 :
200 PRINT"TEAM";X;":"
210 INPUTTE$(X)
215 :
220 PRINT"FAHRZEUGTYP  :"
230 INPUTFT$(X)
235 :
240 PRINT"SCHON BEZAHLT "
250 INPUT"   J";BE$(X)
255 :
260 IFTE$(X)=""ORLEFT$(TE$(X),1)=" "THENX=X-1:RETURN:REM ENDE DER EINGABE
270 :
275 PRINT"ALLES RICHTIG":A$=""
280 INPUT"   JA";A$:IFA$<>"JA"THEN295
290 X=X+1
295 GOTO160
296 :
297 RETURN
300 :
310 :
320 :
330 :
350 REM DRUCKAUSGABE
360 PRINT"MACHEN SIE BITTE DEN  DRUCKER AN.     ":WAIT198,1:GETA$
370 :
380 OPEN4,4,0:REM SIMULIERT AUF EPSON RX 80 DEN COMMODORE-DRUCKER
390 :
400 REM DRUCKAUSGABE
410 PRINT#4,"────────────────────────────────────";
420 PRINT#4,"────────────────────────────────────";
430 PRINT#4
440 PRINT#4
450 PRINT#4
460 PRINT#4
470 PRINT#4
480 OPEN1,4,1:PRINT#1,CHR$(14);"          AUTORALLEY <";RIGHT$(DA$,2);CHR$(20)
CLOSE1
485 PRINT#4
490 PRINT#4
500 PRINT#4
505 PRINT#4
510 PRINT#4
520 PRINT#4
530 PRINT#4
540 PRINT#4,"────────────────────────────────────";
545 PRINT#4,"────────────────────────────────────"
550 PRINT#4
560 PRINT#4,"START  FAHRER / BEIFAHRER  KFZ             ";
570 PRINT#4,"               KM-STAND STARTZEIT"
580 PRINT#4,"NR.    ──────  ────────  ───           ";
590 PRINT#4,"────────────────────────────────────";
610 PRINT#4,"────────────────────────────────────"
620 PRINT#4,"
630 OPEN1,4,1
640 FORK=1TOX
642 IFK<10THENPRINT#1," ";
650 PRINT#1,K;" ";
657 PRINT#1,TE$(K);" AUF ";FT$(K);"   ";BE$(K);
660 :
680 :
690 E=LEN(TE$(K))+LEN(FT$(K))+LEN(BE$(K))+Q
700 P=49-E
710 FORA=1TOP-1:PRINT#4,"-";:NEXTA
720 :
730 :
740 PRINT#4,"└──────────┴──────────┘"
750 :
760 NEXTK
770 :
780 :
790 FORK=1TO5:PRINT#1:NEXTK
800 PRINT#1,CHR$(15);"DIES IST EIN INGO MOLITOR SOFT-IDEE PROGRAMM (C) 1983"
810 PRINT#1:PRINT#1,"BERGHEIM / ERFT DEN ";DA$;CHR$(18)
820 :
840 :
850 CLOSE4:CLOSE1
860 :
870 :
880 RETURN:REM ENDE DRUCKAUSGABE
890 :
900 REM SICHERN DER DATEN
910 :
920 :
930 PRINT"SOLLEN DIE DATEN GE-  SICHERT WERDEN"
940 INPUT"   J";A$:IFA$<>"J"THENRETURN
950 PRINT:PRINT""
960 :
970 :
980 OPEN1,1,2
990 :
1000 PRINT#1,X:PRINT"DATEN WERDEN GESICHERT"
1010 PRINT#1,X
1020 PRINT#1,X:PRINT"EINEN MOMENT BITTE"
1030 PRINT#1,X
1040 PRINT#1,X
1050 :
1060 FORK=1TOX
1070 :
1080 PRINT#1,TE$(K)
1090 PRINT#1,FT$(K)
1100 PRINT#1,BE$(K)
1110 :
1120 NEXTK
1130 CLOSE1
1150 :
1160 RETURN:REM ENDE DER DATENSICHERUNG
1170 :
1180 :
1190 :
1200 REM LADEN DER DATEN
1210 :
1220 :
1230 PRINT"SOLLEN DIE DATEN GE-  LADEN WERDEN"
1240 INPUT"   J";A$:IFA$<>"J"THENRETURN
1260 :
```

Listing 1. Programmteil 1

```
1265 PRINT:PRINT"▒▒"
1270 OPEN2,1,0
1280 :
1290 :
1300 INPUT#2,X:PRINT"DATEN WERDEN GELADEN"
1310 INPUT#2,X
1320 INPUT#2,X:PRINT"EINEN MOMENT BITTE"
1330 INPUT#2,X
1340 INPUT#2,X
1350 :
1360 :
1370 FORK=1TOX
1380 :
1390 INPUT#2,TE$(K)
1400 INPUT#2,FT$(K)
1410 INPUT#2,BE$(K)
1420 :
1430 NEXTK
1440 :
1450 CLOSE2
1460 :
1470 RETURN:REM ENDE DES LADEVORGANGS
1480 :
1490 :
1500 REM BEGINN DES PROGRAMMS
1510 :
1520 PRINT"▒
1530 PRINT"     ┌──────────────┐"
1540 PRINT"     |▒ AUTORALLEY  ▒|"
1550 PRINT"     |  ▒1.PROGRAMM▒ |"
1560 PRINT"     |  ▒┌────────┐▒ |"
1570 PRINT"     | ERFASSEN DER |"
1580 PRINT"     | TEAM'S UND   |"
1590 PRINT"     | FAHRZEUGTYPEN|"
1600 PRINT"▒▒"
1610 PRINT"     └──────────────┘"
1620 PRINT"     ┌──────────────┐"
1625 PRINT"     |▒1=EINGABE    ▒|"
1630 PRINT"     |▒2=DRUCK LISTE▒|"
1640 PRINT"     |▒3=DA. SICHERN▒|"
1645 PRINT"     |▒4=DA. LADEN  ▒|"
1650 PRINT"     |▒5=E N D E    ▒|"
1690 PRINT"▒▒"
1700 PRINT"         ";
1710 INPUT"WAHL     1▒▒▒";A
1720 :
1730 :
1740 IFA>5ORA<1THENPRINT"▒";:GOTO1700
1750 :
1760 ONAGOSUB100,350,900,1200,2000
1770 :
1780 :
1790 :
1800 GOTO1500
2000 PRINT"▒":END
3000 OPEN1,4,0:CMD1
3020 LIST
3030 PRINT#1,FRE(0)

READY.
```

Listing 1. Programmteil 1 (Schluß)

Start
MENUE
1 = Eingabe
2 = Druck Liste
3 = Daten sichern
4 = Daten laden
5 = ENDE
1
2-5
Eingabe der Teams u. Fahrzeuge
Fehler
richtig
2 ? ja Druckausga.
nein
3 ? ja Daten sichern
nein
4 ? ja Daten laden
nein
5 ? ja ENDE
nein

Flußdiagramm des 1. Programmteils

Epson-Druckbefehle

Befehl	Wirkung	Befehl	Wirkung
OPEN 4,4,0	SIMULIERT VC-1525	CHR$(27);F";	EINSCHALTEN DES EP-SON BIT IMAGE-MODE
OPEN 1	EPSON MÖGLICH-KEITEN	CHR$(27);"K-";	
CHR$(14)	BREITSCHRIFT	CHR$(22);	
CHR$(20)	LÖSCHEN BREIT-SCHRIFT	CHR$(0)	
CHR$(15)	SCHMALSCHRIFT	CHR$;(27);	DOPPELDRUCK
CHR$(18)	LÖSCHEN SCHMAL-SCHRIFT	CHR$(27);"H-";	3LÖSCHEN DOP-PELDRUCK
CHR$(27);"E-";	FETTDRUCK	CHR$;(27);	X=1 UNTERSTREICHEN
CHR$(27);"F";	LÖSCHEN FETTDRUCK	CHR$(X)	X=0 UNTERSTREICHEN LÖSCHEN
CHR$(7)	BELL	CHR$(12);	SEITENVORSCHUB
CHR$(27);"N-";CHR$(X)	ÜBERSPRINGEN DER PERFORATION UM X SPALTEN	CMD	ALLE FOLGENDEN AN-GABEN WERDEN ÜBER DEN DRUCKER AUS-GEGEBEN

Steuerzeichen des EPSON-Druckers

So machen's andere

Epson werden mit OPEN 1,4,1 eingeschaltet. Die Epson-Befehle sind im Anschluß an das Listing noch genauer erklärt. Die Jahreszahl des anfangs eingegebenen Datums wird jetzt zur Überschrift der Liste genutzt. Zeile 642 sorgt für rechtsbündige Startnummernausgabe. In Zeile 740 werden Spalten angelegt für die spätere Eintragung von Anfangskilometern und Startzeit. Die fertige Startliste kann man dann am Tag der Rallye mit an den Startplatz nehmen, um die aktuellen Daten dort einzutragen.

3. Daten sichern Zeilen 900 bis 1160

Die Daten werden für den Auswertungsteil gespeichert, entweder auf der Commodore-Datasette oder nach entsprechender Änderung auf Diskette.

4. Daten laden Zeilen 1200 bis 1470

Hier werden die gespeicherten Daten wieder geladen, um eventuelle Änderungen durchzuführen.

5. Ende Zeile 2000

In Zeile 2000 endet das Programm.

Flußdiagramm des 2. Programmteils

2. Programmteil

Nach Abfrage von Datum und Uhrzeit (HHMMSS) wird das Menü gezeigt. Das Datum wird hier nicht vom ersten Programmteil genommen, da ja Anmeldearbeiten und Auswertungsarbeiten an verschiedenen Tagen geschehen.

Der Computer sollte bei der Bearbeitung des zweiten Programmteils am Zielplatz der Rallye stehen, damit schneller auf die neuesten Meldungen reagiert werden kann. Zuerst muß der Menüpunkt »8. Laden alte Daten« gewählt werden, um die Daten aus dem ersten Programmteil wieder abzurufen. Dies geschieht in den Zeilen 6000 bis 6300. Danach kehrt das Programm automatisch zum Menü zurück.

1. Daten ergänzen Zeilen 100 bis 370

In diesem Menüpunkt werden die Daten des ersten Programmteils mit den aktuellen Daten, die am Startplatz in die Liste eingetragen wurden, verknüpft. Hier kann nochmals die Startgeldüberwachung erfolgen und korrigiert werden. Es müssen Start-Kilometer und Start-Zeit eingegeben werden. Sind alle Teams aktualisiert, werden mit

7.A/N Daten Zeilen 4000 bis 4720

die neuen Daten abgespeichert beziehungsweise wieder geladen. Hiermit wird erreicht, daß der Computer nicht stundenlang bis zum Eintreffen der ersten Teilnehmer das Programm halten muß.

Listing 2. Programmteil 2

```
AUTORALLEY 2.TEIL VON INGO MOLITOR

5 POKE36879,25:POKE650,128
10 REM AUSWERTUNG
20 REM DER RALLEY
30 :
40 :
50 :
60 :
70 DIMTE$(100),FT$(100),BE$(100),DI$(100),AM(100),AZ(100),EM(100),KM(100)
75 DIMZ2(100),GZ(100),PK(100),GP(100),G2(100),G3(100),A(100)
80 GOTO3000:REM PROGRAMMANFANG
90 :
100 REM ERGAENZUNG
110 K=1
115 :
120 :
130 :
140 :
150 PRINT"EINGABETEIL"
160 PRINT"────────────"
170 :
180 :
190 :
200 PRINT"TEAM:";K;""
210 PRINTTE$(K)
220 :
230 PRINT"FAHRZEUGTYP:"
240 PRINTFT$(K)
250 :
260 PRINT"BEZAHLT:";BE$(K)
270 INPUT"    J";BE$(K)
275 PRINT"NOCH I. WERTUNG ";DI$(K)
277 INPUT"    J";DI$(K)
280 :
290 INPUT"KM-STAND";AM(K)
295 INPUT"STARTZEIT";AZ(K)
300 :
310 :
320 PRINT"DAS WAR'S !"
330 WAIT198,1:GETA$
340 K=K+1:IFK=X+1ORK>XTHENRETURN
350 :
360 GOTO120
370 REM ENDE DER ERGAENZUNG
380 :
390 :
400 REM ANKUNFT DER TEILNEHMER
410 :
420 :
430 :
440 :
450 PRINT"ANKUNFT"
455 :
460 PRINT"DER TEILNEHMER"
470 PRINT"──────────────"
480 :
490 :
```

2. Ankunft der Fahrer Zeilen 400 bis 960

Eingegeben wird die Startnummer des Teams, das soeben eingetroffen ist. Ist es disqualifiziert, wird das in Zeile 550 erkannt und das Menü erscheint wieder. Ist dieses Team noch in der Wertung, so wird die Abfahrts- und Ankunftszeit gezeigt. Die Ankunftszeit wird von der internen Uhr genommen. Danach erscheint der Anfangskilometerstand und der Endkilometerstand wird abgefragt. In Zeile 610 werden die gefahrenen Kilometer berechnet. In den Zeilen 620 bis 640 werden die Punkte für die benötigte Zeit vergeben. Da Startzeit und Endzeit verrechnet werden, ergibt sich nicht die reale Zeit, sondern ein Wert, der der Punktrechnung zugrundegelegt wird. In der Zeile 635 muß abhängig von der Mindestdauer der Rallye dem Wert der Gesamtzeit ein Wert hinzuaddiert werden (die Gesamtzeit wird von der Mindestdauer abgezogen). Im Programm ist 500 als Wert vorgegeben. In der Zeile 680 werden Strafpunkte

für beispielsweise zu schnelles Fahren wieder abgezogen. Auch dieser Wert muß je nach Dauer der Rallye geändert werden. In Zeile 740 bis 830 können die Punkte für gelöste Aufgaben vergeben werden.

Die erreichten Gesamtpunkte des Teams werden in Zeile 890 errechnet und angezeigt. Stimmen die eingegebenen Daten eines Teams nicht, kann man unter Menüpunkt 2 einfach wieder die Startnummer des Teams eingeben und die Werte korrigieren.

3. Vorläufige Liste Zeilen 1000 bis 1430

In diesem Menüpunkt können vorläufige Ranglisten auf dem Bildschirm angezeigt werden; so kann man zu jedem Zeitpunkt der Rallye den aktuellen Stand der einzelnen Fahrzeuge erkennen und verfolgen. Nachdem alle Fahrer am Zielplatz eingetroffen sind, kann mit Punkt

4. Ausdruck der Liste Zeilen 2000 bis 2910

die endgültige Liste ausgedruckt werden. Mit Open 1,4,1 wird der Epson-Möglichkeiten und mit Open 2,4,0 der Commodore-Drucker angesprochen. In Zeile 2050 wird zu einem Unterprogramm (30000 bis 30900) verzweigt, das grafisch eine Reihe von Autos (oder anderen Symbolen) druckt. Wem diese Möglichkeiten mit seinem Drucker nicht zur Verfügung stehen, der läßt ganz einfach die Zeilen 2050 und 30000 bis 30900 weg. In den Zeilen 2080 bis 2270 werden die Teams mit Rang, Punkten und gefahrenen Kilometern ausgedruckt. Danach springt das Programm automatisch zum Menü zurück.

Für die Sortierung der Daten wurde in Zeile 10000 ff. ein Shell-Sort-Unterprogramm eingebettet. Dieses erlaubt auch die Sortierung großer Datenmengen in relativ kurzer Zeit. Es wurde bewußt keine Quick-Sort-Routine genommen, da diese bei kleineren Datenmengen keinen Zeitvorteil gegenüber Shell-Sort zeigt.

5. Druck der persönlichen Liste Zeilen 7000 bis 7500

Hier wird für jeden Rallyeteilnehmer die Endliste ausgedruckt. Unter der Überschrift erscheint der jeweilige Platz des Teams und das Team wird unterstrichen. Aus Zeitgründen wird hier der Druck der Autoreihe unterlassen. Wem es Spaß macht, der kann natürlich die Zeile 7110 GOSUB 30000 einfügen. Allerdings kostet dieser Spaß dann beim Drucken etwas mehr Zeit.

6. Ende Zeile 41000

Über die Eingabe des Menüpunktes 6 ist das Programm beendet.

Viel Spaß bei der Auswertung der nächsten Orientierungsfahrt. Wem das Eintippen der Programme zu mühselig erscheint, der kann sich gerne an den Autor wenden, und das Programm auf Kassette erhalten. (Ingo Molitor/kg)

```
500 PRINT"TEAM NR.:"
510 INPUTK
520 PRINT"TEAM NR.:";K;"":PRINT"            "
530 PRINT"TEAM:";TE$(K);""
540 :
550 PRINT"NOCH I.WERTUNG ";DI$(K)
555 INPUTDI$(K)
556 IFDI$(K)<>"J"THENTE$(K)=TE$(K)+" DISQUALIFIZIERT":GP(K)=0:G3(K)=0:G2(K)=0:KM
(K)=0
557 IFDI$(K)<>"J"THENRETURN:REM WENN DISQUALIFIZIERT DANN ZURUECK ZUM MENUE
560 PRINT"ABFAHRT (SSMM)"
570 PRINTAZ(K)
580 PRINT"ANKUNFT (SSMM)"
590 Z2(K)=VAL(LEFT$(TI$,4)):PRINTZ2(K)
600 PRINT"ANFANG-KM"
605 PRINTAM(K)
607 PRINT"END-KM"
609 INPUTEM(K)
610 PRINT"GEFAHRENE KM."
612 KM(K)=EM(K)-AM(K):PRINTKM(K)
615 :
620 PRINT"PUNKTE F. ZEIT"
630 GZ(K)=Z2(K)-AZ(K)
635 GZ(K)=-GZ(K)+500:REM MUSS JE NACH LAENGE DER RALLEY VERAENDERT WERDEN
640 PRINTGZ(K);""
650 :
660 REM ZU SCHNELL GEFAHREN
670 REM MUSS JE NACH WUNSCH VERAENDERT WERDEN
680 IFGZ(K)>300THENI=GZ(K)-150:PRINT"STRAFPUNKTE";I:GZ(K)=GZ(K)-I
690 :
700 REM PUNKTE FUER KM MUSS JE NACH WUNSCH VERAENDERT WERDEN
705 PRINT"PUNKTE F. KM"
707 KM(K)=(KM(K)-500)*-1
709 PRINTKM(K)
710 :
720 :
730 :
740 REM PUNKTE FUER GELOESTE AUFGABEN
750 :
760 PRINT"PUNKTE F. AUFGABEN"
770 INPUTPK(K)
780 :
790 :
800 :
810 :
820 INPUT"RICHTIG   J";A$:REM PUNKTE RICHTIG WENN NICHT DANN NEUE PUNKTE
830 IFA$<>"J"ORLEFT$(A$,1)<>"J"THENPRINT"";:GOTO760
840 :
850 :REM GESAMMTPUNKTE
860 :
870 PRINT"GESAMMTPUNKTE"
880 :
890 GP(K)=PK(K)+GZ(K)+KM(K)
900 :
910 PRINT"————";GP(K)
920 :
930 :
940 WAIT198,1:GETA$:RETURN
950 :
960 REM ENDE ANKUNFT  TEILNEHMER
970 :
980 :
990 :
1000 REM VORLAUEFIGE RANGLISTE
1010 :
1020 :
1030 :
1040 PRINT"VORLAUEFIGE LISTE"
1050 PRINT"——————————————————"
1060 :
1070 :
1080 :
1090 REM HILFSVARIABLEN
1100 FORK=1TOX
1120 G2(K)=GP(K):G3(K)=GP(K)
1130 IFLEFT$(DI$(K),1)="N"THENG3(K)=0
1140 NEXTK
1150 :
1160 :
1170 REM SORTIERUNG
1180 :
1190 FORN=1TOX-1
1200 FORK=1TOX-1
1210 :
1220 IFG2(K)>G2(K+1)THENL=G2(K):M=G2(K+1):G2(K)=M:G2(K+1)=L
1230 :
1240 NEXTK,N
1250 :
1260 :
1270 REM ZUORDNUNG DER HILFSVARIABLEN
1280 :
1285 PRINT"PLATZ PUNKTE KFZ "
1290 L=0
1300 FORK=1TOX
1310 :
1320 PRINTX-K+1;G2(K),
1330 :
1340 IFG2(K)=G3(L)THENPRINT" ";FT$(L):L=0:GOTO1380
1350 L=L+1:IFL>XTHENL=0
1360 GOTO1340
1370 :
1380 NEXTK
1390 :
1400 :
```

Listing 2. Programmteil 2 (Fortsetzung)

```
1410 REM ENDE SORTIEREN
1420 WAIT198,1:GETA$
1430 RETURN
1440 :
1450 :
2000 REM AUSDRUCK DER LISTE
2005 REM EROEFFNEN SIMULATIONS-1525 UND EPSON MOEGLICHKEITEN
2010 OPEN1,4,1:OPEN2,4,0
2017 PRINT#1,CHR$(7)
2020 PRINT#1,CHR$(27);"N";CHR$(3);:PRINT#1,CHR$(7);
2025 PRINT#1,CHR$(7);
2030 PRINT#2,"oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooo"
2040 PRINT#1,CHR$(14);"AUSWERTUNG AUTORALLEY VOM ";DA$;CHR$(20)
2045 PRINT#2,"oooooooooooooooooooooooooooooooooooooooooooooooooooooooooooooo
oooooooo"
2050 GOSUB30000:REM DRUCKEN VON AUTOS
2055 PRINT#1
2056 PRINT#1
2057 PRINT#1
2058 PRINT#1
2060 PRINT#1,CHR$(27);"G";
2070 PRINT#1,"RANG  PKT.  KM   FAHRZEUG                TEAM";CHR$(27);"H"
2072 PRINT#2,"────────────────────────────────────────────────────────────
────"
2075 PRINT#1
2080 :
2085 :
2090 GOSUB10000
2100 L=0:H=0
2110 FORK=XTO1STEP-1
2120 L=L+1
2130 IFL<10THENPRINT#1,"  ";L;:GOTO2150
2140 IFL<100THENPRINT#1," ";L;:GOTO2150
2150 PRINT#1," ";
2160 PRINT#1,G3(K);KM(K);"  ";FT$(K);" ";
2180 G=LEN(FT$(K))+8:G=30-G
2185 FORT=1TOG:PRINT#1," ";:NEXTT:REM LEERSTELLEN BIS TEAM DRUCKEN
2190 PRINT#1,CHR$(15);TE$(K);CHR$(18)
2200 H=0
2230 GOTO2270
2260 :
2270 NEXTK
2910 CLOSE1:CLOSE2:RETURN:REM ENDE DRUCKVORGANG
3000 :
3010 REM PROGRAMMVORSTELLUNG DATUM UND UHRZEIT
3020 :
3030 :
3035 PRINT"┌───────────────┐"
3037 PRINT"│               │"
3040 PRINT"│INGO MOLITOR│"
3050 PRINT"│               │"
3060 PRINT"│  SOFTWARE  │"
3070 PRINT"│   IDEE   │"
3080 PRINT"│               │"
3090 PRINT"│PRAESENTIERT│"
3100 PRINT"│               │"
3110 PRINT"│ AUTORALLEY │"
3120 PRINT"│               │"
3130 :
3140 PRINT"│               │"
3142 PRINT"│ AUSWERTUNG  │"
3146 PRINT"│ ──────────  │"
3148 PRINT"│               │"
3150 PRINT"└───────────────┘"
3160 PRINT"(C) BY IMO-SOFT 1983"
3170 FORK=1TO4000:NEXTK
3180 INPUT"TAGESDAT.";DA$
3190 INPUT"UHRZEIT    HHMMSS";TI$
3200 :
3210 :
3220 REM ANFANG MENUE
3230 :
3240 PRINT"MENUE"
3250 PRINT"         "
3260 :
3270 :
3280 PRINT"1 =DA.ERGAENZEN"
3290 PRINT"2 =ANKUNFT D. FAHRER"
3300 PRINT"3 =VORLAUEFIGE LISTE":REM AUF BILDSCHIRM
3310 PRINT"4 =AUSDRUCK D. LISTE"
3320 PRINT"5 =DRUCK D. P. LISTE"
3325 PRINT"6 =E N D E        "
3330 PRINT"7 =A/N DATEN      "
3340 REM A/N DATEN FUER UNTERBRECHUNG DES PROGRAMMS BZW. WIEDERAUFNAHME
3350 PRINT"8 =LADEN ALTE DATEN":REM DATEN DES 1.PROGRAMMS
3360 :
3370 INPUT"WAHL    2";A
3380 :
3385 ONAGOSUB100,400,1000,2000,7000,41000,4000,6000
3390 GOTO3200
3400 :
3410 :
4000 REM LADEN SAVEN DATEN
4010 PRINT">L< ADEN"
4020 PRINT"ODER "
4030 PRINT">S< AVEN"
4040 PRINT"DATEN ?"
4050 :
4060 INPUT"WAHL    *";W$:IFW$="*"THENRETURN
4070 IFW$<>"S"THENIFW$<>"L"THENRETURN
4080 :
4090 :
4100 IFW$="S"THENGOTO4510
4110 :
4120 PRINT"OK.DATEN WERDEN GE-     LADEN"
4130 :
4140 OPEN1,1,0
4150 :
4160 INPUT#1,X:PRINT"EINEN MOMENT BITTE"
4170 INPUT#1,X
4180 INPUT#1,X
4190 INPUT#1,X
4200 INPUT#1,X
4210 :
4220 FORK=1TOX
4230 INPUT#1,TE$(K)
4240 INPUT#1,FT$(K)
4250 INPUT#1,BE$(K)
4260 INPUT#1,DI$(K)
4270 INPUT#1,AM(K)
4280 INPUT#1,AZ(K)
4290 :
4300 NEXT
4310 CLOSE1
4320 RETURN:REM ENDE LADEN DATEN ZURUECK ZU MENUE
4510 :
4520 PRINT"OK.DATEN WERDEN GE-     SAVED"
4540 OPEN1,1,2
4550 :
4560 PRINT#1,X:PRINT"EINEN MOMENT BITTE"
4570 PRINT#1,X
4580 PRINT#1,X
4590 PRINT#1,X
4600 PRINT#1,X
4610 :
4620 FORK=1TOX
4630 PRINT#1,TE$(K)
4640 PRINT#1,FT$(K)
4650 PRINT#1,BE$(K)
4660 PRINT#1,DI$(K)
4670 PRINT#1,AM(K)
4680 PRINT#1,AZ(K)
4690 :
4700 NEXT
4710 CLOSE1
4720 RETURN:REM ENDE SAVEN DATEN ZURUECK ZU MENUE
4990 :
6000 REM LADEN DATEN DES 1.PROGRAMMS
6010 :
6020 :
6030 :
6040 PRINT"SOLLEN ALTE DATEN GE- LADEN WERDEN"
6050 :
6060 INPUT"     N";A$
6070 :
6080 IFA$="N"ORLEFT$(A$,1)<>"J"THENRETURN
6090 :
6095 :
6100 OPEN1,1,0
6110 :
6120 INPUT#1,X:PRINT"EINEN MOMENT BITTE"
6130 INPUT#1,X
6140 INPUT#1,X
6150 INPUT#1,X
6160 INPUT#1,X
6170 :
6180 :
6190 FORK=1TOX
6200 :
6210 INPUT#1,TE$(K)
6220 INPUT#1,FT$(K)
6230 INPUT#1,BE$(K)
6265 NEXTK
6270 :
6280 CLOSE1
6290 :
6300 RETURN:REM ENDE LADEN DATEN DES 1.PROGRAMMS
     ZURUECK ZUM MENUE
7000 :
7010 REM DRUCK DER PERSOENLICHEN LISTE
7020 :
7030 R=0
7040 FORK=XTO1STEP-1
7042 R=R+1
7045 OPEN1,4,1:OPEN2,4,0:REM SIMULIEREN DES
     VC-1525 UND EPSON MOEGLICHKEITEN
7050 PRINT#1,CHR$(7)
7060 PRINT#1,CHR$(27);"N";CHR$(3);:PRINT#1,CHR$(7);
7070 PRINT#1,CHR$(7);
7080 PRINT#2,"oooooooooooooooooooooooooooooooooooooo
oooooooo"
7090 PRINT#1,CHR$(14);"AUSWERTUNG AUTORALLEY VOM ";
     DA$;CHR$(20)
7100 PRINT#2,"oooooooooooooooooooooooooooooooooooooo
oooooooo"
7110 :
7120 PRINT#1
7125 PRINT#1
7130 PRINT#2,"                                 "
7132 PRINT#2,"                            PLATZ";R
7134 PRINT#2,"                                 "
7136 PRINT#1
7140 PRINT#1
7150 PRINT#1,CHR$(27);"G";
```


Listing 2. Programmteil 2 (Fortsetzung)

```
7160 PRINT#1,"RANG  PKT.  KM   FAHRZEUG                TEAM";CHR$(27);"H"
7170 PRINT#2,"──────────────────────────────────────────────────────────────────
──────"
7180 PRINT#1
7200 GOSUB10000
7220 FORY=XTO1STEP-1
7230 L=L+1
7240 IFY=KTHENPRINT#1,CHR$(27);"-";CHR$(1);:REM JEWEILGES TEAM WIRD UNTERSTRICHE
N
7250 IFL<10THENPRINT#1,"  ";L;:GOTO7270
7260 IFL<100THENPRINT#1," ";L;:GOTO7270
7270 PRINT#1," ";
7280 PRINT#1,G3(Y);KM(Y);" ";FT$(Y);" ";
7300 G=LEN(FT$(Y))+8:G=30-G
7310 FORT=1TOG:PRINT#1," ";:NEXTT
7320 PRINT#1,CHR$(15);TE$(Y);CHR$(18)
7350 :
7360 :
7370 IFY=KTHENPRINT#1,CHR$(27);"-";CHR$(0);
7380 NEXTY
7385 :
7810 PRINT#1,CHR$(27);"E"
7815 PRINT#1,"DIE RENNLEITUNG BEDANKT SICH FUER IHR FAIRES MITFAHREN !";CHR$(27)
;"F"
7820 REM PRINT#1,CHR$(12)
7840 CLOSE1:CLOSE2
7850 NEXTK
7900 RETURN:REM ENDE DES AUSDRUCKS PERSOENLICHE LISTEN
10000 REM SHELL-METZNER-SOTIERROUTINE
10010 :
10020 :
10030 J6=X
10040 J6=INT(J6/2)
10050 IF(J6=0)THEN10490
10060 J2=X-J6
10070 FORJ=1TOJ2
10080 I=J
10090 J3=I+J6
10100 IF(G3(I)<=G3(J3))THEN10160
10110 H1=G3(I):H2=KM(I):H3$=FT$(I):H4$=TE$(I)
10120 G3(I)=G3(J3):KM(I)=KM(J3):FT$(I)=FT$(J3):TE$(I)=TE$(J3)
10130 G3(J3)=H1:KM(J3)=H2:FT$(J3)=H3$:TE$(J3)=H4$
10140 I=I-J6
10150 IF(I>0)THEN10090
10160 NEXTJ
10170 GOTO10040
10490 RETURN:REM ENDE SHELL-SORT
30000 REM AUTOS DRUCKEN
30010 REM IN EPSON BIT IMAGE SCHREIBWEISE
30015 PRINT#1,CHR$(27);"F";
30025 FORT=1TO20
30030 PRINT#1,CHR$(27);"K";CHR$(22);CHR$(0);
30040 FORN=1TO22
30045 READD
30050 PRINT#1,CHR$(D);
30060 NEXTN
30065 RESTORE:NEXTT
30800 DATA24,56,56,56,60,126,158,156,152,248,248,152,152,152,156,158,126,60,56,2
4,0,0
30900 RESTORE:RETURN
40000 OPEN1,4,0:CMD1:LIST:PRINT#1
41000 PRINT"[illegible]";:END

READY.
```


Listing 2. Programmteil 2 (Schluß)

Programm zur Tastenabfrage aus dem Tastaturpuffer

```
405 REM **** C 64 PROGRAMM 4 *********
406 REM TASTENABFRAGE AUS T.-PUFFER *
407 REM*******************************
410 PRINT CHR$(147)
420 POKE 198,0
430 A=PEEK(631)
440 IF A=133 THEN POKE 53280,6: POKE 53281,7
450 IF A=137 THEN POKE 53380,5: POKE 53281,2
460 IF A=134 THEN POKE 53280,1: POKE 53281,1
470 IF A=64 THEN POKE 53280,3: POKE 53281,1
480 GOTO 420
```

```
405 REM***VC 20 PROGRAMM 4 **********
406 REM TASTENABFRAGE AUS T.-PUFFER *
407 REM ******************************
410 PRINT CHR$(147)
420 POKE 198,0
430 A=PEEK(631)
440 IF A=133 THEN POKE 36879,126
450 IF A=137 THEN POKE 36879,45
460 IF A=134 THEN POKE 36879,24
470 IF A=64 THEN POKE 36879,27
480 GOTO 420
```

Sie haben natürlich gemerkt, daß die Abfrage der f3-Taste kombiniert mit CTRL aus dem Tastaturpuffer nicht geht. Der Grund dafür ist, daß diese Kombination keinen eigenen ASCII-Code hat. Die so laut gepriesenen 32 Funktionstasten sind also nur bei einer Abfrage der Speicherzellen 203 und 253 möglich. In Zeile 460 des Programms 4 habe ich daher reumütig die f3-Taste allein verwendet.

Diese beiden Programme gehören als Lösung zu einer Aufgabe die im Beitrag »Alle Tasten-, Zeichen- und Steuercodes« gestellt wurde.

Wohin mit dem Heimcomputer?

Eine berechtigte Frage, besonders bei beengten Platzverhältnissen. Eine zwar nicht neue, aber originelle und praktische Lösung eines Lesers stellen wir hier vor.

Bild 2. Bei Gebrauch »Klappe auf — Monitor ran — los geht's«

Bild 3. Das Ganze aus einem anderen Blickwinkel

Bild 1. Bei Nichtgebrauch steht die große Arbeitsfläche zur Verfügung (der Monitor ist »schwebend« in die Ecke verbannt)

Bild 4. Auch der Drucker hat seinen Platz gefunden. Man beachte die praktische Lösung mit dem Papierkorb.

Die Benutzung des Wohnzimmertisches oder gar des meist viel besser geeigneten Eßzimmertisches stößt in der Regel auf heftigen Widerstand. Der Küchentisch, ebenfalls nicht zu verachten, schließt sich durch seine ungünstige Lage selbst aus. Wer hat schon in der Küche einen Fernseher?

Apropos Fernseher — man kommt auf die Dauer (des lieben Friedens willen) ohne eigenes Gerät nicht aus. Welche Ehefrau duldet schon die recht ansehnlichen Bilder mit »PEEKs«, »POKEs« oder gar »SYNTAX ERROR«, wenn sich gerade so Fieslinge wie »J.R.« oder »Alexis Carrington« auf dem Bildschirm bewegen.

Arbeitsplatz-Rationalität — nicht nur ein Schlagwort

Der Heimcomputer sollte an einem Ort stehen, wo er geschützt, weitgehend unsichtbar (nach einem mißlungenen Programm sollte man die Genugtuung haben, ihn verbannen zu können, ohne ihn gleich aus dem Fenster zu werfen), aber jederzeit ohne großen Aufwand (auspacken, aufbauen, Kabel verlegen und anschließen) betriebsbereit ist.

Außerdem sollte er bei Nichtgebrauch möglichst wenig Platz beanspruchen, sonst heißt es wieder: »Dein Hobby nimmt ja die halbe Wohnung ein.« Es reicht ja, wenn man sich diesen Vorwurf während seiner »wissenschaftlich-schöpferischen« Tätigkeit anhören muß. Hier meine Lösung! Zugegeben — ich bin in der glücklichen Situation, ein Zimmerchen (Neubau-Kinderzimmer — die armen Kleinen) fast für mich allein belegen zu dürfen. Außer Gästebett und Kleiderschrank, und das bei acht Quadratmetern, ist der Rest mein Reich. Ach so, da die Tür bei der »Größe« des Zimmers zweckmäßigerweise nach innen aufgeht, muß ich auf einen weiteren Quadratmeter verzichten.

Würde ich nun die Geräte auf einer Tischplatte aufbauen, so hätte ich zwar einen beachtlichen Arbeitsplatz, der aber ausschließlich dem Heimcomputer gewidmet wäre (oder Aufwand wie oben). Also gab es für mich nur die in den Bildern 1 bis 4 dargestellte Lösung.

(Dieter Wienands)

Das Lösungswort lautet:

PROGRAMMIER
WETTBEWERB

Sicherlich haben Sie es schon mehr als einmal erlebt, daß Ihr Computer Ihnen Rätsel aufgab — nämlich immer dann, wenn beim Starten eines neu eingetippten Programms mal wieder gar nichts so lief, wie es laufen sollte. Warum sollte der Computer also nicht auch einmal im wörtlichen Sinne Rätsel aufgeben?

1000 Mark zu gewinnen

Gesucht wird ein Programm für den C 64 oder den VC 20, das Kreuzworträtsel in einem Format von 20 x 20 Kästchen erstellt. Dabei soll auf ein von Ihnen erstelltes »Lexikon« von minimal 100 Wörtern zurückgegriffen werden. Zusätzlich muß es allerdings möglich sein, über die Tastatur eine Anzahl von Begriffen einzugeben, die dann auch im Rätsel auftauchen sollten und im Lexikon abgespeichert werden. Das Einsetzen der ersten Begriffe ins Rätsel soll rein zufällig erfolgen, damit bei mehreren Programmläufen auch jedesmal verschiedene Rätsel erzeugt werden.

Beurteilt wird Ihr Programm nach der Geschwindigkeit, in der das Kreuzworträtsel erstellt wird, wobei die Anzahl der schwarzen Kästchen möglichst gering sein sollte. Das beste Kreuzworträtselprogramm ist uns einen Tausender wert.

Schicken Sie Ihr Programm mit ausführlicher Beschreibung und Probeausdruck eines erzeugten Kreuzworträtsels an: Redaktion 64'er, Stichwort Kreuzworträtsel, Hans-Pinsel-Str. 2, 8013 Haar bei München.

Wir suchen die Anwendung des Monats

Anwendung des Monats, was ist das? Nun, Sie haben einen Commodore 64 oder einen VC 20 und versuchen diesen irgendwie sinnvoll einzusetzen. Unter einer sinnvollen Anwendung versteht die 64'er Redaktion alles, was beispielsweise Programme im häuslichen Bereich bewirken. Es kann sich dabei um die Berechnung der Benzinkosten für Ihren Wagen handeln, um ein eigenes Textverarbeitungsprogramm gehen, sich um die Verwaltung Ihrer Tiefkühltruhe drehen oder ein ausgeklügeltes Telefon- und Adreßregister sein.

Setzen Sie Ihren VC 20/C 64 mehr oder weniger beruflich ein? Auch, oder vor allem, das ist eine sinnvolle Anwendung. Sie führen die Lohn- und Gehaltsabrechnung, Ihre Lagerverwaltung, die Bestellungen auf einem Commodore-Heimcomputer durch? So spezielle Anwendungen wie die Berechnung der Statik von selbstgezimmerten Regalen, von Klimadiagrammen oder Vokabellernprogrammen für den Schulunterricht oder die Zinsberechnung bei Krediten sind ebenfalls Themen, die mehr als konkurrenzfähig sind.

Uns ist die Anwendung des Monats

500 Mark

wert

Schreiben Sie uns, was Sie mit Ihrem Computer machen: Redaktion 64'er, Aktion: Anwendung des Monats, Hans-Pinsel-Str. 2, 8013 Haar bei München

Einmal im Monat gibt es die

SUPERCHANCE

Diese nicht einmalige Gelegenheit sollten Sie nutzen. Wie? Schicken Sie uns Ihr bestes, selbst erstelltes Programm. Bei der Art des Programms sind wir nicht wählerisch.
Sie haben ein sehr gutes (Schieß-, Knobel-, Denk-, Action-, Abenteuer-) Spiel geschrieben: einschicken!
Sie verfügen über ein komfortables Disketten-Kopier-(Sortier-)Programm mit einigen außergewöhnlichen Leistungsmerkmalen: einschicken!
Sie haben das Basic um einige sinnvolle Befehle erweitert: einschicken!
Sie arbeiten mit einem selbsterstellten Textverarbeitungsprogramm, einer eigenen Tabellenkalkulation, einem semiprofessionellen Datenverwaltungsprogramm: einschikken!
Sie zeichnen und konstruieren mit einem selbsterstellten Programm in hochauflösender Grafik: einschicken!
Wir freuen uns über jeden Beitrag und honorieren mit bis zu

2000 Mark

für das Listing des Monats

Aus den besten Listings, die veröffentlicht werden, sucht die 64'er Redaktion einmal im Monat das »Listing des Monats« aus. Alle Listings, die im 64'er abgedruckt sind, werden mit 100 bis 300 Mark honoriert. Die genaue Vorgehensweise beim Einsenden von Listings ist in »Wie schicke ich meine Programme ein?« (letzte Ausgabe) beschreiben:

Schicken Sie Ihr Listing an: Redaktion 64'er, Superchance: Listing des Monats, Hans-Pinsel-Str. 2, 8013 Haar bei München

Erklärung der Steuerzeichen

Da sich immer wieder Schwierigkeiten bei der Identifizierung der Steuerzeichen in Listings ergeben, sind hier alle Steuerzeichen für die Commodore-Drucker VC 1515, VC 1526, MPS 801 (dem Nachfolgemodell des VC 1525) und für den Printer/Plotter 1520 angegeben.

Funktion	VC 1515		VC 1526		MPS 801		VC 1520	
CLR	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	s	S
HOME	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	S	s
REVERS ON	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	R	r
REVERS OF	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	f	F
INST	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	t	T
STOP	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	C	c
RECHTS	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	]	△
LINKS	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	△	]
UNTEN	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	Q	q
OBEN	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	q	Q
SCHWARZ	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	p	P
WEISS	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	E	e
ROT	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	£	—
TUERKIS	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	□	←
VIOLETT	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	—	£
GRUEN	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	↑	π
BLAU	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	←	□
GELB	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	π	↑
ORANGE	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	a	A
BRAUN	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	u	U
HELLROT	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	v	V
GRAU 1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	w	W
GRAU 2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	x	X
HELLGRUEN	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	y	Y
HELLBLAU	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	z	Z
GRAU 3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	|	[
F1	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	e	E
F2	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	i	I
F3	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	f	F
F4	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	j	J
F5	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	g	G
F6	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	k	K
F7	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	h	H
F8	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	[illegible]	l	L

Großbuchstaben/Grafikmodus

Groß-/Kleinbuchstaben

Die Steuerzeichen des VC 1520 sind in den Listings unterstrichen, da der Printer/Plotter keine reversen Zeichen darstellen kann.

Hier gibt es Clubs

Es gibt nun auch einen C 64-Club in Hannover, der sich vorgenommen hat, nicht ein Club von vielen zu werden. Er will nicht den Fehler machen, mit deutscher Gründlichkeit einen Vorsitzenden, einen Beisitzer sowie einen Kassenwart zu wählen. Clubstatuten soll ebenso ein Fremdwort sein wie Vereinsmeierei. Nur eines gibt's auch: einen Mitgliedsbeitrag. Dafür wird auch einiges geboten. Ein Clubmagazin erscheint zirka 10mal jährlich und bringt alles rund um den C 64, angefangen vom Softwaretest über neue Listings, Kurse in Maschinensprache bis hin zu etlichen interessanten Tips und Tricks für die Praxis. Es ist auch ein Raum angemietet, in dem es bei Clubtreffen zum Erfahrungsaustausch oder einfach nur zum Klönen kommt; außerdem gibt's ein Servicetelefon, an dem sich jedes Mitglied Rat bei Problemen holen kann. Außerdem wird eine Softwarebank aufgebaut, die allen Mitgliedern zur Verfügung steht. Durch Programm- und Listingtausch von selbsterstellter Software wird sie automatisch ständig erweitert. Groß- und Sammelbestellungen werden gestartet, um durch Mengenrabatte für die Mitglieder günstiger einkaufen zu können. Kontaktadresse: C-64 User Club Germany, Hildesheimer Str. 388, 3000 Hannover 81, Tel. 0511/863037

Hardcopy auf Tastendruck?

Ich möchte mit dem VC 20 eine Hardcopy machen, Drucker-Type 1525. Dazu möchte ich eine der Programmfunktions-Tasten f1-f8 des VC 20 benutzen. Es geht also darum, jederzeit eine Hardcopy zu machen, egal was auf dem Bildschirm ist, ohne jedesmal den Befehl zur Hardcopy im Programm zu schreiben. (Bildschirmanzeige) = drücke Taste f1-f8 = Hardcopy. Was muß ich tun?

Klaus von der Beck

Eine solche Hardcopy auf Tastendruck in jeder Situation wäre sicherlich eine wünschenswerte Eigenschaft, ist aber auf der Basic-Ebene nicht zu realisieren. Eine mögliche Lösung per Maschinenprogramm ist durch Eingriff in die Tastaturabfrage des Computers möglich. Der Interruptvektor muß hierzu so »verbogen« werden, daß vor der eigentlichen Tastaturabfrage ein Sprung in eine zu schreibende Maschinensprache-Routine erfolgt, welche die Funktionstasten abfragt und gegebenenfalls ein Hardcopy-Unterprogramm aufruft.

Inserentenverzeichnis

Impressum

Herausgeber: Carl-Franz von Quadt, Otmar Weber

Chefredakteur: Michael M. Pauly (py)
Stellv. Chefredakteur: Michael Scharfenberger (sc)
Redakteure: aa = Albert Absmeier (130), ev = Volker Everts (130), kg = Karin Gößlinghoff (269), gk = Georg Klinge (278), rg = Christian Rogge (278)
Redaktionsassistenz: Dagmar Zednik (237)
Fotografie: Janos Feister, Titelfoto: Alex Kempkens

Layout: Leo Eder (Ltg.), Willi Gründl, Walter Höß, Cornelia Weber

Auslandsrepräsentation:
Schweiz: Markt&Technik Vertriebs AG, Alpenstrasse 14, CH-6300 Zug, Tel. 042-223155/56, Telex: 862329 mut ch
USA: M&T Publishing, 2464 Embarcadero Way, Palo Alto, CA 94303; Tel. 001-4240600; Telex 752351

Manuskripteinsendungen: Manuskripte und Programmlistings werden gerne von der Redaktion angenommen. Mit der Einsendung von Manuskripten und Listings gibt der Verfasser die Zustimmung zum Abdruck und zur Vervielfältigung der Programmlistings auf Datenträger. Honorare nach Vereinbarung. Für unverlangt eingesandte Manuskripte und Listings wird keine Haftung übernommen.

Herstellung: Klaus Buck (180), Leo Eder (181)

Anzeigenleitung: Peter Schrödel (156)
Anzeigenverkauf: Alfred Reeb (211)

Anzeigenverwaltung und Disposition: Michaela Hörl (171)

Anzeigenformate: 1/1-Seite ist 266 Millimeter hoch und 185 Millimeter breit (3 Spalten à 58 mm oder 4 Spalten à 43 Millimeter). Vollformat 297 x 210 Millimeter. Beilagen und Beihefter siehe Anzeigenpreisliste.

Anzeigenpreise: Es gilt die Anzeigenpreisliste Nr. 1 vom 1. März 1984.
Anzeigengrundpreise: 1/1 Seite sw: DM 7400,-. Farbzuschlag: erste und zweite Zusatzfarbe aus Europaskala je DM 1000,-. Vierfarbzuschlag DM 3000,-. Plazierung innerhalb der redaktionellen Beiträge: Mindestgröße 1/1-Seite
Anzeigen im Einkaufs-Magazin: Die ermäßigten Preise im Einkaufs-Magazin gelten nur innerhalb des geschlossenen Anzeigenteils, der ohne redaktionelle Beiträge ist. 1/1-Seite sw: DM 5400,-. Farbzuschlag: erste und zweite Zusatzfarbe aus Europaskala je DM 1000,-. Vierfarbzuschlag DM 3000,-. **Anzeigen in der Fundgrube: Private Kleinanzeigen** mit maximal 5 Zeilen Text DM 5,- je Anzeige.
Gewerbliche Kleinanzeigen: DM 10,- je Zeile Text.
Auf alle Anzeigenpreise wird die gesetzliche MwSt jeweils zugerechnet.

Vertriebsleitung, Werbung: Hans Hörl (114)

Vertrieb Handelsauflage: Inland (Groß-, Einzel- und Bahnhofsbuchhandel) sowie Österreich und Schweiz: Pegasus Buch- und Zeitschriften-Vertriebs GmbH, Plieninger Straße 100, 7000 Stuttgart 80 (Möhringen), Telefon (0711) 72004-0

Erscheinungsweise: 64'er, Magazin für Computerfans, erscheint monatlich, Mitte des Vormonats.

Bezugsmöglichkeiten: Leser-Service: Telefon 089/4613-238. Bestellungen nimmt der Verlag oder jede Buchhandlung entgegen. Das Abonnement verlängert sich zu den dann jeweils gültigen Bedingungen um ein Jahr, wenn es nicht zwei Monate vor Ablauf schriftlich gekündigt wird.

Bezugspreise: Das Einzelheft kostet DM 6,-. Der Abonnementspreis beträgt im Inland DM 72,- pro Jahr für 12 Ausgaben. Darin enthalten sind die gesetzliche Mehrwertsteuer und die Zustellgebühren. Der Abonnementspreis erhöht sich um DM 18,- für die Zustellung im Ausland, für die Luftpostzustellung in Ländergruppe 1 (z.B. USA) um DM 38,-, in Ländergruppe 2 (z.B. Hongkong) um DM 58,-, in Ländergruppe 3 (z.B. Australien) um DM 68,-.

Druck: St. Otto-Verlag, Bamberg

Verantwortlich: Für redaktionellen Teil: Michael M. Pauly.
Für Anzeigen: Peter Schrödel.

Vorstand: Carl-Franz von Quadt, Otmar Weber

Anschrift für Verlag, Redaktion, Vertrieb, Anzeigenverwaltung und alle Verantwortlichen:
Markt&Technik Verlag Aktiengesellschaft, Hans-Pinsel-Straße 2, 8013 Haar bei München, Telefon 089/4613-0, Telex 5-22052

Mitteilung gem. Bayerischem Pressegesetz: Aktionäre, die mehr als 25% des Kapitals halten: Otmar Weber, Ingenieur, München; Carl-Franz von Quadt, Betriebswirt, München. Aufsichtsrat: Dr. Robert Dissmann (Vorsitzender), Karl-Heinz Fanselow, Hans-Jochen Wolf.

Telefon-Durchwahl im Verlag:

Wählen Sie direkt: Per Durchwahl erreichen Sie alle Abteilungen direkt. Sie wählen 089-4613 und dann die Nummer, die in Klammern hinter dem jeweiligen Namen angegeben ist.

In der nächsten Ausgabe vom 18. Mai lesen Sie:

VORSCHAU

Test: Textverarbeitungs-programme

Eine Anwendung, die immer mehr in »Mode« kommt, ist die Textverarbeitung mit dem Commodore 64. Das Angebot an guten und weniger guten Programmen ist mittlerweile fast unüberschaubar geworden. Wir haben uns die besten deutschen und amerikanischen Textverarbeitungsprogramme herausgepickt und getestet: SM Text 64, Blitztext, Vizawrite und Wordpro 3+. Eine Marktübersicht der gängigsten Textverarbeitungsprogramme wird das Bild abrunden.

Commodore 116: Der VC 20-Nachfolger?

Zu dem Erfolg der Commodore-Computer trägt in nicht geringem Maße die ausgezeichnete Tastatur bei. Auch der neue Commodore 116 wird damit ausgestattet sein.

Grafische Darstellung statt Zahlenfriedhof

Übersichtliche Grafik statt Zahlenkolonnen: Dieses Programm für den C 64 stellt wahlweise Balken-, Kuchen- oder konventionelle Strichdiagramme dar.

Rund um die Datasette

Wir haben für Sie getestet, was die Original-Commodore-Datasette, Datasetten von anderen Herstellern und Interfaces für handelsübliche Kassettenrecorder leisten oder was sie nur versprechen.

Exbasic Level II

In der Reihe der Basic- und Toolkit-Erweiterungen für den C 64 steht das Exbasic Level II von Interface Age auf den ersten Plätzen. Wer Exbasic kennengelernt hat, benutzt es bald bei jeder Programmiertätigkeit. Was ist so interessant, was kann Exbasic Level II? Ist es besser als Simons Basic oder ergänzen sich diese beiden Erweiterungen?

Epson-Drucker am Commodore 64

Die Epson-Drucker FX/RX 80 zeichnen sich durch ein gutes Preis/Leistungsverhältnis aus. Was liegt da näher, als der Wunsch, sie auch an den Commodore 64 anzuschließen. Wir zeigen Ihnen, was Sie dabei beachten müssen und welche Probleme es geben kann.

Zwei VC 20 gekoppelt

Können zwei VC 20 Daten austauschen und gar mit- und gegeneinander spielen? Es geht, sogar relativ einfach. Wir geben eine Bauanleitung für ein Interface und liefern auch die nötige Treibersoftware dazu.

Listings

— Anwendungen des Monats: C 64 führt Lehrerkalender
— Listing des Monats: Underground — ein abenteuerliches Spiel für den VC 20
— Analyse von Maschinenprogrammen leichtgemacht: Logischer Disassembler für C 64/VC 20 verfolgt Programmierlogik
— C 64-Programme, die sich selbst starten: der Trick mit dem Autostart
— Anwendungen: Vokabellernprogramm, Morsetrainer, Textverarbeitung
— Spiele: Hot Wheels, ein Motorradrennen und Ghost Manor, ein Abenteuerspiel
— außerdem natürlich wieder viele Tips und Tricks für den C 64 und VC 20